COMPUTER LISTING APPENDIX

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.*;

/**
 * This is the main container for all WML documents.  It is directly analogous to a WML file ↙
   containing
 * any number of cards and card elements.
 */
public class WMLTagDocument extends TagDocument
{

    public final static String DEFAULT_XML_LANG = "en-us";

    public final static String XML_HEADER = "<?xml version=\"1.0\"?>";
    public static String DOC_TYPE = "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN      ↙
        \" \"http://www.wapforum.org/DTD/wml_1.1.xml\">";

    Head head;
    Template template;
    Card card;

    /**Creates a standard WMLTagDocument (&lt;wml&gt;&lt;/wml&gt;)
     */
    public WMLTagDocument() {
        super("wml","text/vnd.wap.wml");
    }

    /**Creates a WMLTagDocument with the specified 'xml:lang' attribute.  In general,
     * you should use the no arg constructor.
     *
     * @param xml_lang the specified xml:lang type.
     */
    public WMLTagDocument(String xml_lang) {
        this();
        getRoot().addAttribute("xml:lang",xml_lang);
    }

    /**Set's the Head tag for this deck.
    *
    */
    public void setHead(Head head)  throws InvalidTagException {
        this.head = head;
        resetChildren();
    }

    /**Set's a Template for this deck.
    *
    */
    public void setTemplate(Template template)  throws InvalidTagException {
        this.template = template;
        resetChildren();
    }

    public void setCard(Card card) throws InvalidTagException {
        this.card = card;
        resetChildren();
    }

    /***Adds a Card to this document.
     *
     * @param card A Card to be added to this document.
     */
    public void addCard(Card card) {
          try { addChild(card); }
          catch(InvalidTagException e) {
            e.printStackTrace();
          }
        }
    /***This clears all of the children for this Card.
     */
    private void resetChildren() throws InvalidTagException {

        getRoot().getChildren().removeAllElements();

        if (head != null)
            addChild(head);

        if (template != null)
            addChild(template);

        if (card != null)
            addChild(card);
    }

    /***This renders the entire WMLTagDocument.
     *
     * @return String the document rendered to a String.
     */
    public String render() {

        StringBuffer output = new StringBuffer();
        output.append(XML_HEADER + "\n");
        output.append(DOC_TYPE + "\n\n");

        output.append(super.render());

        return output.toString();
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.*;

import java.util.Enumeration;

/**
* The basic WMLTag which all other tag classes in this package subclass
*/
public class WMLTag extends Tag
{
    /**
    * @param name the tag text to use <"name">
    * @param closingTag indicates if this is a standalone tag, or if it has an accompanying ↙
        closing tag
    */
    public WMLTag (String name, boolean closingTag)
    {
        super(name,closingTag);
    }

    /**
    * @param name the tag text to use <"name">
    */
    public WMLTag (String name)
    {
        super(name);
    }

    /**
    * called by the render() method to render the start tag
    */
    protected String renderOpenTag() {
        StringBuffer output = new StringBuffer();

        //render self open
        output.append("<");
        output.append(getName());

        Enumeration eAttribs = getAttributes().elements();

        while(eAttribs.hasMoreElements())
            output.append(" " + ((Attribute)eAttribs.nextElement()).render());

        if (!isClosedTag())
            output.append("/");

        output.append(">");

        return output.toString();
    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.AbstractCharacterEncoder;

/**
 * A utility class that reformats characters, such as ", <,
 * >, ', &, $, etc. which are reserved for
 * use by WML/WAP, and that must been encoded in all uses.
 *
 * This class is a singleton so it has a private constructor
 *
 * @author kleemax
 */
public class WMLCharacterEncoder extends AbstractCharacterEncoder
{
///////////////////////////////////////////////////////////
// CONSTANTS
    private final static String[] ENCODED_STRINGS = { "&amp;",
                                                      "&quot;",
                                                      "&lt;",
                                                      "&gt;",
                                                      "&apos;",
                                                      "$$",
                                                      " ",
                                                      " ",
                                                      "(TM)",
                                                      "&apos;" };

    private final static char[] CHARS_TO_ENCODE = { '&',    // AMPERSAND
                                                    '"',    // QUOTE
                                                    '<',    // LESS THAN
                                                    '>',    // GREATER THAN
                                                    39,     // the ' apostrophy
                                                    '$',    // DOLLARSIGN
                                                    '\n',   // NEW LINE
                                                    '\r',   // LINE FEED
                                                    8482,   // TRADEMARK
                                                    180 };  // FORWARD APOSTROPHY

    // UNUSED
    private final static String SOFT_HYPHEN = "&shy;";
    private final static String NON_BREAKING_SPACE = "&nbsp;";

///////////////////////////////////////////////////////////
// VARIABLES
    private static WMLCharacterEncoder s_Singleton = null;

///////////////////////////////////////////////////////////
// PRIVATE CONSTRUCTOR
    private WMLCharacterEncoder() {
        // Singleton.  No constructor
    }

///////////////////////////////////////////////////////////
// ABSTRACT METHODS OVERRIDDEN FROM SUPERCLASS

    /**Allows the subclass to specify the list of characters that should
     * be encoded.  Note that this list should have the same number of
     * elements, with each element in the same position, as the array
     * that is returned from getEncodedStrings().
     *
     * @return A list of special characters which should be encoded in
     * the current markup language.
     */
    protected char[] getCharactersToEncode() {
        return CHARS_TO_ENCODE;
    }

    /**Allows the subclass to specify the list of strings that should
```

```
     * be used as escape sequences for each character that should be
     * encoded.  Note that this list should have the same number of
     * elements, with each element in the same position, as the array
     * that is returned from getCharactersToEncode().
     *
     * @return A list of strings which are valid escape sequences in
     * the current markup language.
     */
    protected String[] getEncodedStrings() {
        return ENCODED_STRINGS;
    }

//////////////////////////////////////////////////////////
// METHODS
    public static String clipAndEncode(String inText, int inStart,
                        int inLength, String inTail, int[] outLengthUsed)    {
        if (s_Singleton == null) {
            s_Singleton = new WMLCharacterEncoder();
        }

        return s_Singleton.clipAndEncodeWithTail(inText, inStart, inLength, inTail,
            outLengthUsed);
    }

}
```

```
package com.thinairapps.tag.wml;

import java.util.Enumeration;
import java.util.Properties;

import java.net.URLEncoder;

/**This is a convenience class for building URLs (href Strings) suitable for use within  ↙
    various markup languages.
*
*/
public class URLBuilder
{
    public static String DELIMETER = "&amp;";
    public static String NAME_VALUE_DELIMETER = "=";

    private final static String RND_KEY = "trnd";
    private final static String DEFAULT_BASE_URL = "?";

    /**Builds a URL suitable for use within a WML deck.
    *
    *@param baseURL the domain name, path, etc. as needed.  Everything required before the  ↙
        '?'.
    *@param props a Properties object containing the name value pairs of the HTTP URL  ↙
        parameters.
    *@param addRandom indicates if a random parameter should be added in the form, 'trnd  ↙
        ="1234"'.  This
    * is used as a means to force browsers not to use cached pages.
    */
    public static String buildWapUrl (String baseUrl, Properties props, boolean addRandom)
    {

        StringBuffer url = new StringBuffer();

        if (baseUrl != null)
        {
            url.append (baseUrl);

            if (!baseUrl.endsWith(DEFAULT_BASE_URL))
                url.append (DEFAULT_BASE_URL);
        }
        else
            url.append (DEFAULT_BASE_URL);

        Enumeration enum = props.keys();

        String key = null, value = null;

        while (enum.hasMoreElements())
        {
            key = (String)enum.nextElement();

            value = props.getProperty (key);

            if (!value.startsWith("$"))
                value = URLEncoder.encode (value);

            url.append (key);
            url.append (NAME_VALUE_DELIMETER);
            url.append (value);

            if (enum.hasMoreElements())
                url.append (DELIMETER);
        }

        if (addRandom)
        {
            url.append (DELIMETER);
            url.append (RND_KEY);
            url.append (NAME_VALUE_DELIMETER);
            url.append (genRnd());
        }

        return url.toString();

    }

    private static String genRnd ()
    {
        String rnd = new java.util.Date().getTime() +"";

        //trim the random number to reduce URL size
        int length = rnd.length();
        if (length >=4)
            rnd = rnd.substring(length-4,length);
        return rnd;
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;timer&gt; element provides a method for invoking a task automatically after some   ↙
    period of user inactivity. Any task or user action that activates the card starts the   ↙
    timer, and executing any task element stops it. You can only associate one task per timer↙
    , and you can only define one timer per card.
*/
public class Timer extends WMLTag
{
    /**
    * @param name The name of the variable in which the device stores the timer value. If the↙
        variable has no value when the timer is initialized, the device sets it to the value ↙
        specified for the default attribute. The device sets this variable to either the     ↙
        current timer value when the user exits the card or 0 if the timer expires.
    * @param value A string specifying the value for the variable specified by the key        ↙
        attribute. You must specify <timer> values in units of 1/10 seconds--so, for example ↙
        , a value of 100 equals 10 seconds. Specifying a value of 0 disables the timer.
    */
    public Timer(String name,int value) {
        super("timer",false);
        addAttribute("name",name);
        addAttribute("value",value * 10 + "");
    }

    /**
    * @param value A string specifying the value for the variable specified by the key        ↙
        attribute. You must specify <timer> values in units of 1/10 seconds--so, for example ↙
        , a value of 100 equals 10 seconds. Specifying a value of 0 disables the timer.
    */
    public Timer(int value) {
        this("default",value);
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**This is a convenience class for formatting Text in a WMLdeck.
*
*/
public class TextStyle extends WMLTag
{
    public final static String PLAIN = "";
    public final static String BOLD = "b";
    public final static String BIG = "big";
    public final static String EMPHASIZE = "em";
    public final static String ITALIC = "i";
    public final static String SMALL = "small";
    public final static String STRONG = "strong";
    public final static String UNDERLINED = "u";

    String style = PLAIN;

    /**
    *@param style one of the String constants defined in this class.
    */
    public TextStyle(String style) {
        super("",true);
        this.style = style;
    }

    /**
    *@param style one of the String constants defined in this class.
    *@param child a child tag to be formatted in the given syle.
    */
    public TextStyle(String style,WMLTag child)  throws InvalidTagException {
        super("",true);
        this.style = style;
        addChild(child);
    }

    protected String renderOpenTag() {
        if (style != PLAIN)
            return "<" + style + ">";
        else
            return "";
    }

    protected String renderCloseTag() {
        if (style != PLAIN)
            return "</" + style + ">";
        else
            return "";
    }

}
```

```
package com.thinairapps.tag.wml;

/**
 * This is a simple class that allows you to insert an arbitrary String
 * into a WML Deck.
 */
public class Text extends WMLTag
{
    /***Create a new Text Object with the given String
     */
    public Text(String text) {
        super(text,false);
    }

    /***Get the String you used to define this Text Object.
     */
    public String getTextOnly() {
        return getName();
    }

    /***Return the Sting defined for this Text Object
     */
    public String render() {
        return getName();
    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
 * Element that defines deck-level event bindings (i.e. characteristics that apply to all
   cards in the deck)
 */
public class Template extends WMLTag
{
    public Template() {
        super("template");
    }

    /**
     * @param onEnterForward specifies the URL to open if the user navigates to a card
         through a &lt;go&gt; task
     * @param onEnterBackward specifies the URL to open if the user navigates to a card
         through a &lt;prev&gt; task
     * @param onTimer specifies the URL to open if the &lt;timer&gt; element expires
     */
    public Template(String onEnterForward,String onEnterBackward,String onTimer) {
        this();
        addAttribute("onenterforward",onEnterForward);
        addAttribute("onenterbackward",onEnterBackward);
        addAttribute("ontimer",onTimer);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Do || child instanceof OnEvent)
            super.addChild(child);
        else
            throw new InvalidTagException("Template only supports Do and OnEvent child
                tags");
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The basic Task type which is subclassed by other classes (&lt;go&gt;, &lt;do&gt;)
*/
public abstract class Task extends WMLTag
{
    public Task (String name, boolean closed)
    {
        super(name,closed);
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;tr&gt; element is used as a container to hold a single table row. Table rows may be
    empty,
* in other words, all cells are empty.
*/
public class TableRow extends WMLTag
{
    public TableRow() {
        super("tr");
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof TableCell)
            super.addChild(child);
        else
            throw new InvalidTagException("TableRow only supports TableCell child tags");
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;td&gt; element is used as a container to hold a single table cell data within a ↙
    table row.
* Table data cells may be empty. The user agent should do a best effort to deal with multiple↙
    line
* data cells that may result from using images or line breaks.
*/
public class TableCell extends WMLTag
{
    public TableCell() {
        super("td");
    }

    /*Constructs a TableCell with the given child tag.
    */
    public TableCell(WMLTag child) throws InvalidTagException {
        this();
        addChild(child);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Text || child instanceof Image || child instanceof Anchor)
            super.addChild(child);
        else
            throw new InvalidTagException("TableCell only supports Text, Image, and Anchor ↙
                children tags");
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;table&gt; element allows you to specify columnar format. WML tables are similar to ↙
    HTML
* tables but with fewer capabilities. When defining a table, you have to declare the number ↙
    of columns,
* followed by some content. The content can include empty rows and columns.
*/
public class Table extends WMLTag
{
    public final static String ALIGN_LEFT = "left";
    public final static String ALIGN_CENTER = "center";
    public final static String ALIGN_RIGHT = "right";

    public Table() {
        super("table");
    }

    /**
    * @param title Specifies a label for the table.
    * @param align (ALIGN_LEFT, ALIGN_CENTER, ALIGN_RIGHT) Specifies text alignment relative ↙
        to the column.
    * If you do not specify the align attribute, the text is automatically left-aligned.
    *
    * @param columns Specifies the number of columns for the row set. Specifying a zero value↙
        for
    * this attribute is not allowed.
    */
    public Table(String title,String align,int columns)
    {
        this();
        addAttribute("title",title);
        addAttribute("align",align);
        addAttribute("columns","" + columns);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof TableRow)
            super.addChild(child);
        else
            throw new InvalidTagException("Table only supports TableRow child tags");
    }

}
```

```
package com.thinairapps.tag.wml;

/**The &lt;spawn&gt; element defines a spawn task, indicating the creation of a child context
* and invocation of a URL in that child context. If the URL names a WML card or deck, the
    card
* is displayed, and the URL becomes the basis for a new history stack in the child context.
    See
* specific WML browser documentation for more details.
*
*/
public class Spawn extends WMLTag
{
    public Spawn ()
    {
        super ("spawn", true);
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* A widget for generating a card with a single text-type input.  For details on the ↙
    constructors, see the constructors for
* Card.
*/
public class SingleInputCard extends Card
{

    public SingleInputCard() {
        super();
    }

    public SingleInputCard(String id) {
        super(id);
    }

    public SingleInputCard(String id,String title) {
        super(id,title);
    }

    public SingleInputCard(String id,String title,boolean newContext) {
        super(id,title,newContext);
    }

    public SingleInputCard(String id,String title,boolean newContext,boolean ordered) {
        super(id,title,newContext,ordered);
    }

    /**
    * @param href the url to which the results of the select input are to be submited
    * @param label the label to use with the input field
    * @param name the name of the input field
    * @param format the format mask to use with the input field
    */
    public void buildCard(String href,String label,String name,String format)
    {

        Do do1 = new Do(Do.TYPE_ACCEPT,new Go(href,false));
        addChild(do1);

        Paragraph p = new Paragraph();
        p.addChild(new Text(label));
        Input input = new Input(name);
        input.setFormat(format);
        p.addChild(input);
        addChild(p);

    }
    /**
     * Build a Card that uses the POST method
    * @param href the url to which the results of the select input are to be submited
    * @param label the label to use with the input field
    * @param name the name of the input field
    * @param format the format mask to use with the input field
    * @param extraFields array of &lt;postfield&gt; elements
    */

    public void buildPostCard (String href,String label,String name,String format, PostField ↙
        [] extraFields)
    {
        Go goa = new Go(href, true, Go.METHOD_POST);
        goa.addChild(new PostField(name,"$"+name));

        if (extraFields != null)
        {
            for (int i = 0; i < extraFields.length; i++)
                goa.addChild(extraFields[i]);

        }

        Do dew = new Do(Do.TYPE_ACCEPT,goa);
        addChild(dew);

        Paragraph p = new Paragraph();
        p.addChild(new Text(label));
        Input input = new Input(name);
        input.setFormat(format);
        p.addChild(input);
        addChild(p);

    }

}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;setvar&gt; element sets a variable to a specified value when the device executes a ↙
   &lt;go&gt;,
* &lt;prev&gt;, &lt;spawn&gt;, or &lt;refresh&gt; task.
*/
public class SetVar extends WMLTag
{
    public SetVar (String name, String value)
    {
        super("setvar",false);
        addAttribute ("name",name);
        addAttribute ("value",value);
    }
}
```

09872376.060601

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* A widget for generating a card with a Select list input.  For details on the constructors, ↙
    see the constructors for
* Card.
*/
public class SelectInputCard extends Card
{

    public SelectInputCard() {
        super();
    }

    public SelectInputCard(String id) {
        super(id);
    }

    public SelectInputCard(String id,String title) {
        super(id,title);
    }

    public SelectInputCard(String id,String title,boolean newContext) {
        super(id,title,newContext);
    }

    public SelectInputCard(String id,String title,boolean newContext,boolean ordered) {
        super(id,title,newContext,ordered);
    }

    /**
    * @param href the url to which the results of the select input are to be submited
    * @param label the text label for this input
    * @param name the name of the select tag
    * @param optionVals name/value pairs for all of the options
    * @param align the text alignment to use for the display (Paragraph.LEFT, etc)
    * @param mode the text wrap mode to use for the display (Paragraph.NO_WRAP)
    */
    public void buildCard(String href,String label,String name,String[][] optionVals,String ↙
        align,String mode) throws InvalidTagException {

        Option[] options = new Option[optionVals.length];

        for (int i = 0; i < optionVals.length; i++) {
            options[i] = new Option(optionVals[i][1]);
            options[i].setLabel(optionVals[i][0]);
        }

        buildCard(href,label,name,options,align,mode);
    }

    /**
    * @param href the url to which the results of the select input are to be submited
    * @param label the text label for this input
    * @param name the name of the select tag
    * @param options the set of &lt;option&gt; tags to use for this select
    * @param align the text alignment to use for the display (Paragraph.LEFT, etc)
    * @param mode the text wrap mode to use for the display (Paragraph.NO_WRAP)
    */
    public void buildCard(String href,String label,String name,Option[] options,String align,↙
        String mode) throws InvalidTagException {

        Do do1 = new Do(Do.TYPE_ACCEPT,new Go(href,false));
        addChild(do1);

        Paragraph pHeader = new Paragraph(align,Paragraph.MODE_WRAP);
        pHeader.addChild(new Text(label));
        addChild(pHeader);

        Paragraph p = new Paragraph(align,mode);

        Select select = new Select("",name,false);
        Option cOpt = null;

        for (int i = 0; i < options.length; i++)
            select.addOption(options[i]);

        p.addChild(select);
        addChild(p);
    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;select&gt; tag element specifies a list of options from which the user can choose. ↙
    You can
* specify either single- or multiple-choice &lt;select&gt; tag elements.
*/
public class Select extends WMLTag
{
    public Select() {
        super("select",true);
    }

    /**
    * @param name The name of the variable in which the device stores the value(s) associated↙
        with
    * the option(s) chosen by the user. The value associated with each option comes from the
    * &lt;option&gt; tag element value attribute.
    * @param title  Specifies a brief label for the &lt;select&gt; list. Some devices use the↙
        label
    * as a title when displaying the &lt;select&gt; content. Others might use it as a label    ↙
        for a user
    * interface mechanism that lets the user navigate to the &lt;select&gt; content. For      ↙
        example, if a
    * device cannot display all card content on one screen and ordered="true" (see Order      ↙
        options), the
    * UP.Browser uses the title to identify this select list on a summary-level menu.
    * @param multiple Specifies whether the user can select multiple items.
    */
    public Select(String title,String name,boolean multiple) {
        this();
        addAttribute("title",title);
        addAttribute("name",name);
        addAttribute("multiple","" + multiple);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Option || child instanceof OptGroup)
            super.addChild(child);
        else
            throw new InvalidTagException("Select only supports OptGroup or Option children  ↙
                tags");
    }

    public void addOption(Option option) {
        try { addChild(option); }
        catch(InvalidTagException e) {}
    }

    /**
    *@param defaultValue A string specifying the default value(s) for the variable specified ↙
        by
    * the name attribute.
    */
    public void setDefaultValue(String defaultValue) {
        addAttribute("value",defaultValue); //TODO what is the proper attribute? "value"?
    }

    /**
    * @param iname Identical to the name attribute except for the following: The specified
    * variable stores the index value(s) associated with the option(s) chosen by the user.   ↙
        The
    * index value associated with each option comes from its position in the <i>select</i>   ↙
        list,
    * starting with 1. If the user has not selected an option, the index value is 0. The     ↙
        default
    * value is specified by the ivalue attribute.
    * @param ivalue Identical to the default attribute except for the following: The ↙
       specified
    * string contains the default index value(s) for the variable specified by the iname ↙
       attribute.
    */
    public void setINameAndIValue(String iName,String iValue) {
        addAttribute("iname",iName);
        addAttribute("ivalue",iValue);
    }

    public void setTabIndex(int tabIndex) {
        addAttribute("tabIndex","" + tabIndex);
    }

}
```

```java
package com.thinairapps.tag.wml;

/**
 * Used to reset all variables within a deck's current context.
 */
public class Reset extends WMLTag
{
    public Reset ()
    {
        super ("Reset",false);
    }
}
```

```
package com.thinairapps.tag.wml;

/**
 * A utility class that reformats characters, such as ", <, >, ', &, $, etc. which are
   reserved for
 * use by WML/WAP, and that must been encoded in all uses
 *
 * @author kleemax
 */
public class ReservedCharacter {
    public final static String QUOTE = "&quot;";
    public final static String LESS_THAN = "&lt;";
    public final static String GREATER_THAN = "&gt;";
    public final static String APOSTROPHY = "&apos;";
    public final static String AMPERSAND = "&amp;";
    public final static String DOLLARSIGN = "$$";
    public final static String SOFT_HYPHEN = "&shy;";
    public final static String NON_BREAKING_SPACE = "&nbsp;";
    public final static String TRADEMARK = "(TM)";

    public final static int ESCSEQ_LENGTH = 6;

    /**Processes inText, returning it's substring from inStart so that it is no
     * longer than inLength, and has inTail tacked on if it was longer.  All of
     * the special characters in inText are reformatted to Markup Language Escape
     * Sequences.  If inTail is not null, then it is tacked onto the end of strings
     * which are too long, to show that the string is incomplete.  Empty String ""
     * is returned if inText is null or "", if inStart is invalid, or if inLength
     * is <= 0.
     *
     * @param inText the raw text that we will clip and encode
     *
     * @param inStart the starting offset in inText from which we should start the
     * returned string.  Nothing before inStart will be included in the return string.
     *
     * @param inLength the approximate number of characters that we would like our
     * returned string to be.  the returned string may be slightly more than inLength
     * if it ends in an escape sequence.
     *
     * @param inTail if inTail is not null or "", then it will be tacked onto the
     * end of the returned string if we do not all of inText from inStart on.
     * BEWARE: inTail does NOT get URL encoded and Escaped.  Make sure that it doesn't
     * contain special characters.
     *
     * @param outLengthUsed is a call-by-reference parameter.  Pass in a non-null
     * int array at least 1 long if you wish to know the number of characters from
     * inText that were actually used in building the return String.
     * outLengthUsed[0] will contain the number of chars from inText which were
     * actually used or escaped in the returned string.  This is NOT the length of
     * the returned String.  Instead, it is the length of inText that was used in
     * building the returned String.
     *
     * @return a string that starts from start and is no longer than length, with
     * all of the special characters escaped out.
     *
     * TODO: in the future, this method should look into having a parameter that
     * lets us break on words, rather than just breaking on length.  (i.e.: do it
     * word by word, so we don't have incomplete words.)
     */
    public static String clipAndEncode(String inText, int inStart, int inLength,
                                       String inTail, int[] outLengthUsed)  {
        // Deal with the case when we get bad parameters in
        if (inText == null || inText.length() == 0 || inStart < 0 ||
            inLength <= 0 || inStart >= inText.length()) {
            if (outLengthUsed != null) {
                outLengthUsed[0] = 0;
            }
            return "";
        }
        // Local variables
        StringBuffer buff = new StringBuffer(inLength);
        int length = inText.length(),
            used = 0,
            charsVisited = 0;
        char current;
        boolean usingTail = false;

        // Check to see if we need to use the tail
        if (inTail != null && inTail.length() > 0) {
            // add the tail if it is specified
            usingTail = true;
            inLength -= inTail.length();
        }

        // Loop around char by char, inserting the chars into the buffer.
        while (charsVisited < inLength && inStart < length) {
            used++;
            current = inText.charAt(inStart);

            switch (current) {
            case '&':
                buff.append(AMPERSAND);
                charsVisited += AMPERSAND.length();
                break;
            case '"':
                buff.append(QUOTE);
                charsVisited += QUOTE.length();
                break;
            case '<':
                buff.append(LESS_THAN);
                charsVisited += LESS_THAN.length();
                break;
            case '>':
                buff.append(GREATER_THAN);
                charsVisited += GREATER_THAN.length();
                break;
            case 39:    //the ' apostrophy
                buff.append(APOSTROPHY);
                charsVisited += APOSTROPHY.length();
                break;
            case '$':
                buff.append(DOLLARSIGN);
                charsVisited += DOLLARSIGN.length();
                break;
            case '\n':
                buff.append(' ');
                charsVisited++;
                break;
            case '\r':
                buff.append(' ');
                charsVisited++;
                break;
            case 8482:  //the trademark symbol
                buff.append(TRADEMARK);
                charsVisited += TRADEMARK.length();
                break;
            case 180:   // the forward apostrophe
                buff.append(APOSTROPHY);
                charsVisited += APOSTROPHY.length();
                break;
            default:
                buff.append(current);
                charsVisited++;
                break;
            }
            inStart++;
        }
        // Add the tail if it was called for, and we didn't iterate over the
        // whole length
        if (usingTail && inStart < length) {
            buff.append(inTail);
        }

        // Now report the length used if it is asked for (by passing
        // in a non-null outLengthUsed
        if (outLengthUsed != null && outLengthUsed.length > 0) {
            outLengthUsed[0] = used;
        }

        return buff.toString();
            // Should this be .trim() trimmed?
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;refresh&gt; element is a task element that instructs the device to refresh the
    specified card variables. The device also refreshes the display if any of those variables
    are currently shown.
*/
public class Refresh extends Task
{
    public Refresh() {
        super("refresh", true);
    }

    public Refresh(boolean closing) {
        super("refresh",closing);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (this.isClosedTag())
            if (child instanceof SetVar)
                super.addChild(child);
            else
                throw new InvalidTagException("Refresh only supports SetVar children tags");
        else
            throw new InvalidTagException("Tag must be a closing tag to support children");
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;prev&gt; element is a task element that instructs the device to remove the current ↙
    URL from
* the history stack and open the previous URL. If no previous URL exists on the history stack↙
    , specifying <prev> has no effect.
*/
public class Prev extends Task
{
    /*Constructs a new Prev tag.
    *
    */
    public Prev() {
        super("prev",true);
    }

    /*Constructs a new opening or closing Prev tag, depending on the 'closing' parameter.
    *
    */
    public Prev(boolean closing) {
        super("prev",closing);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (this.isClosedTag())
            if (child instanceof SetVar)
                super.addChild(child);
            else
                throw new InvalidTagException("Prev only supports SetVar children tags");
        else
            throw new InvalidTagException("Tag must be a closing tag to support children");
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;postfield&gt; element defines name/value pairs that are passed to the HTTP server  ↙
    receiving
* the &lt;go&gt; request. See the &lt;go&gt; element for an example of the &lt;postfield&gt; ↙
    element's use
* in WML.
*/
public class PostField extends WMLTag
{
    /**
    *@param name A label that identifies the field.
    *@param value A string specifying the default value for the variable specified by the    ↙
        value attribute.
    */
    public PostField(String name,String value) {
        super("postfield",false);
        addAttribute("name",name);
        addAttribute("value",value);
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* A WML Widget used to create a card with a labeled text input using a Password "*" mask.   ↙
   For constructor details,
* see Card.
*/
public class PasswordInputCard extends Card
{

    public PasswordInputCard()
    {
        super();
    }

    public PasswordInputCard (String id)
    {
        super(id);
    }

    public PasswordInputCard (String id,String title)
    {
        super(id,title);
    }

    public PasswordInputCard (String id, String title, boolean newContext)
    {
        super(id,title,newContext);
    }

    public PasswordInputCard (String id, String title, boolean newContext, boolean ordered)
    {
        super(id,title,newContext,ordered);
    }

    /**
    *@param href the link to submit the input to
    *@param label the text label to display next to the input
    *@param the name of the field
    *@param the format to use for the field (*M,n)
    */
    public void buildCard (String href, String label, String name, String format) throws   ↙
        InvalidTagException
    {

        Do do1 = new Do(Do.TYPE_ACCEPT,new Go(href,false));
        addChild(do1);

        Paragraph p = new Paragraph();
        p.addChild(new Text(label));
        Input input = new Input(name);
        input.setType("password");
        input.setFormat(format);
        input.setValue("");         // parksan (10/17/2k) - added this to remove caching
        p.addChild(input);
        addChild(p);
    }

    /**
    * @param href the link to submit the input to
    * @param label the text label to display next to the input
    * @param name the name of the field
    * @param format the format to use for the field (*M,n)
    * @param extraFields an array of PostFields posted to the server along with the user's   ↙
        form input
    */
    public void buildPostCard (String href, String label, String name, String format,   ↙
        PostField[] extraFields)
    {
        Go goa = new Go(href, true, Go.METHOD_POST);
        goa.addChild(new PostField(name,"$"+name));

        if (extraFields != null)
        {
            for (int i = 0; i < extraFields.length; i++)
                goa.addChild(extraFields[i]);

        }

        Do dew = new Do(Do.TYPE_ACCEPT,goa);
        addChild(dew);

        Paragraph p = new Paragraph();
        p.addChild(new Text(label));
        Input input = new Input(name);
        input.setType("password");
        input.setFormat(format);
        input.setValue("");          // parksan (10/17/2k) - added this to remove caching
        p.addChild(input);
        addChild(p);

    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;p&gt; tag element specifies a new paragraph and has alignment and line-wrapping ↙
    attributes.
*/
public class Paragraph extends WMLTag
{
    public final static String ALIGN_LEFT = "left";
    public final static String ALIGN_CENTER = "center";
    public final static String ALIGN_RIGHT = "right";

    public final static String MODE_WRAP = "wrap";
    public final static String MODE_NOWRAP = "nowrap";

    public Paragraph()
    {
        super("p");
    }

    /**
    * @param align (ALIGN_LEFT, ALIGN_CENTER, ALIGN_RIGHT) Specifies line alignment relative ↙
        to the display area. Specifying &amp;lt;p&amp;gt; without the align attribute resets ↙
        the line to left alignment.
    * @param mode (MODE_WRAP, MODE_NOWRAP) Specifies text wrapping mode to use. If you ↙
        specify nowrap, the device uses another mechanism, such as horizontal scrolling, to ↙
        display long lines to the user. The device continues to use the mode you specify ↙
        until you specify a &amp;lt;p&amp;gt; element with the other mode.
    */
    public Paragraph (String align, String mode)
    {
        this();
        addAttribute("align",align);
        addAttribute("mode",mode);
    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;option&gt; tag element specifies a particular choice within a <i>select</i> tag ↙
    element.
*/
public class Option extends WMLTag
{
    public Option() {
        super("option");
    }

    /**
    * @param value Specifies the value to assign to the variable defined in the <i>select</i ↙
        > tag
    * element name attribute if the user selects the option (see example). If you specify a
    * variable reference, the device evaluates the reference before setting the name variable↙
        .
    */
    public Option(String value) {
        this();
        addAttribute("value",value);
    }

    /**
    * @param title A label that identifies the option. The UP.Browser uses the title as the
    * ACCEPT key label when the user selects the option. To ensure compatibility on a wide
    * range of devices, label should be a maximum of five characters.
    * @param value Specifies the value to assign to the variable defined in the <i>select</i ↙
        > tag
    * element name attribute if the user selects the option (see example). If you specify a
    * variable reference, the device evaluates the reference before setting the name variable↙
        .
    */
    public Option(String title,String value) {
        this(value);
        addAttribute("title",title);

    }

    /**
     * @param title A label that identifies the option. The UP.Browser uses the title as the
     * ACCEPT key label when the user selects the option. To ensure compatibility on a wide
     * range of devices, label should be a maximum of five characters.
     * @param value Specifies the value to assign to the variable defined in the <i>select</i↙
        > tag
     * element name attribute if the user selects the option (see example). If you specify a
     * variable reference, the device evaluates the reference before setting the name       ↙
        variable.
     * @param label the text to be used as the displayable option
     */
    public Option(String title, String value, String label) {
        this(title, value);
        setLabel(label);
    }


    /**
    * @param onPick Specifies the URL to open if the user selects the option (or deselects it
    * if the <i>select</i> element allows multiple choices). This attribute represents an
    * abbreviated form of the &lt;onevent&gt; element.
    */
    public void setOnPick(String onPick) {
        addAttribute("onpick",onPick);
    }
    /**
    *@param label the text to be used as the displayable option
    */
    public void setLabel(String label) throws InvalidTagException {
        addChild(new Text(label));
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Text || child instanceof OnEvent)
            super.addChild(child);
        else
            throw new InvalidTagException("Option only supports Text or OnEvent children     ↙
                tags");
    }
}
```

```java
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;optgroup&gt; element allows you to group multiple &lt;option&gt; (or nested
* &lt;optgroup&gt;) elements within a card. Creating option groups lets you specify control
* information about how the device should present the card content.
*/
public class OptGroup extends WMLTag
{
    public OptGroup() {
        super("optgroup");
    }

    /**
    * @param title Specifies a brief label for the &lt;optgroup&gt; group. Some devices use
    * the label as a title when displaying the &lt;optgroup&gt; content. Others might use it
    * as a label for a user interface mechanism that lets the user navigate to the
    * &lt;optgroup&gt; content.
    */
    public OptGroup(String title) {
        super("optgroup");
        addAttribute("title",title);
    }

    /*
    *@param child a Tag of type Option or OptGroup
    */
    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof OptGroup || child instanceof Option)
            super.addChild(child);
        else
            throw new InvalidTagException("OptGroup only supports OptGroup or Option children↙
                tags");
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;onevent&gt; element associates a state transition, or intrinsic event, with a task ↙
    . When the
* intrinsic event occurs, the device performs the associated <onevent> task.
*/
public class OnEvent extends WMLTag
{
    public final static String TYPE_ON_PICK = "onpick";
    public final static String TYPE_ON_ENTER_FORWARD = "onenterforward";
    public final static String TYPE_ON_ENTER_BACKWARD = "onenterbackward";
    public final static String TYPE_ON_TIMER = "ontimer";

    /**
    * @param type Identifies the intrinsic event that triggers the specified <onevent> task  ↙
        (see
    * descriptions below). If a card-level <onevent> element (i.e. defined within a <card>  ↙
        element)
    * has the same type as a deck-level <onevent> element (i.e. defined within a <template>  ↙
        element),
    * the card-level binding overrides the deck-level binding.
    */
    public OnEvent(String type) {
        super("onevent");
        addAttribute("type",type);
    }

    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Task)
            super.addChild(child);
        else
            throw new InvalidTagException("OnEvent only supports Task children tags");
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;noop&gt; element is a task element that instructs the device to do nothing, i.e     ↙
    . "no operation."
* This element is useful for overriding deck-level <do> elements, called shadowing (see the   ↙
    UP.SDK
* Developer's Guide for more information on shadowing).
*/
public class NoOperation extends Task
{
    public NoOperation() {
        super("noop",false);
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* An insertable space (&nbsp;)
*/
public class NonBreakingSpace extends Text
{
    public NonBreakingSpace(int number) {
        super("");

        for (int i = 0; i < number; i++)
            setName(getName() + "&nbsp;");
    }
}
```

```java
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* A WML widget which creates a card with multiple <i>input</i> tags. The WAP
* Browser will either display this as a single card with "popup" input
* fields, or as multiple cards which the user steps through.
*/
public class MultipleInputCard extends Card
{

    public MultipleInputCard() {
        super();
    }

    /**Constructs a new MultipleInputCard with the given name (id).
     *
     * @param id specifies a name for the card
     */
    public MultipleInputCard(String id) {
        super(id);
    }

    /**Constructs a new MultipleInputCard with the given name ('id') and label ('title')  ↙
        attributes.
     *
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     */
    public MultipleInputCard(String id,String title) {
        super(id,title);
    }

    /**Constructs a new MultipleInputCard with the given name (id), label (title) and  ↙
         'newcontext' attributes.
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     * @param newContext specifies whether the device should initialize the context whenever ↙
        the user naviages to the card through a &lt;go/&gt; task
     */
    public MultipleInputCard(String id,String title,boolean newContext) {
        super(id,title,newContext);
    }

    /**Constructs a new MultipleInputCard with the given name ('id'), label  ↙
        ('title'), 'newcontext' attributes,
     * and 'ordered' attributes.
     *
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     * @param newContext specifies whether the device should initialize the context whenever ↙
        the user naviages to the card through a &lt;go/&gt; task
     * @param ordered specifies the organization of card content
     */
    public MultipleInputCard(String id,String title,boolean newContext,boolean ordered) {
        super(id,title,newContext,ordered);
    }

    /**
    * @param href the url to pass the result of the form input to
    * @param acceptLabel the text label to use on the accept button ("save", "submit", etc)
    * @param method the method to use to submit the form data to the server ("POST","GET")
    */
    public void buildCard (String href,String acceptLabel,LabeledInput[] inputs,String  ↙
        method) throws InvalidTagException
    {
        buildCard(null,href,acceptLabel,inputs,method);
    }
    public void buildCard (String header, String href,String acceptLabel,LabeledInput[]      ↙
        inputs,String method) throws InvalidTagException
    {
        buildCard (header, href, acceptLabel, inputs, method, null);
    }

    /**
    * @param header text to be displayed at the top of the form
    * @param href the url to pass the result of the form input to
    * @param acceptLabel the text label to use on the accept button ("save", "submit", etc)
    * @param method the method to use to submit the form data to the server ("POST","GET")
    */
    public void buildCard (String header, String href, String acceptLabel, LabeledInput[]     ↙
        inputs, String method, PostField[] hiddenFields) throws InvalidTagException
    {
        Paragraph p = new Paragraph();

        boolean closingGo = false;

        if (method.equals(Go.METHOD_POST))
            closingGo = true;

        Go go1 = new Go(href, closingGo, method);
        Do do1 = new Do(Do.TYPE_ACCEPT,go1);

        do1.addAttribute("label",acceptLabel);

        if (method.equals(Go.METHOD_POST))
        {
            for (int i = 0; i < inputs.length; i++)
                go1.addChild(new PostField(inputs[i].getInputName(),"$" + inputs[i].        ↙
                    getInputName()));

            if (hiddenFields != null && hiddenFields.length > 0)
                 for (int i = 0; i < hiddenFields.length; i++)
                    go1.addChild(hiddenFields[i]);
        }

        p.addChild(do1);

        if (header != null)
        {
            p.addChild(new Text(header));
            p.addChild(new Break());
        }
        for (int i = 0; i < inputs.length; i++)
            p.addChild(inputs[i]);

        addChild(p);
    }

}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;meta&gt; element provides meta information for a WML deck. This element is
    specified
* within the deck header along with any access control information for the deck (for more
* information, see &lt;access&gt; element and &lt;head&gt; element). Note that not all
    devices
* support every meta information type.
*/
public class Meta extends WMLTag
{
    public final static String PROPERTY_NAME = "name";
    public final static String PROPERTY_HTTP_EQUIV = "http-equiv";
    public final static String PROPERTY_USER_AGENT = "user-agent";

    public final static String PROPERTY_USER_AGENT_MARKABLE = "vnd.up.markable";
    public final static String PROPERTY_USER_AGENT_BOOKMARK = "vnd.up.bookmark";

    /**
    * @param propertyType the attribute name to use with this Meta tag (i.e. name, http-equiv
        , etc)
    * @param propertyValue the value for the attribute indicated in propertyType
    * @param content Specifies the metadata value associated with the property attribute.
    */
    public Meta(String propertyType,String propertyValue,String content) {
        super("meta",false);
        addAttribute(propertyType,propertyValue);
        addAttribute("content",content);
    }

    /**
    * @param forua (true | false)  Specifies that the author intended the property to reach
        the
    * user agent. If forua="false", an intermediate agent must remove the &lt;meta&gt;
        element
    * before the document is sent to the client. If the value is "true", the metadata of the
    * element must be delivered to the user agent. The method of delivery may vary. For
        example,
    * http-equiv metadata may be delivered using HTTP or WSP headers.
    */
    public void setForua(boolean forua)
    {
        addAttribute("forua","" + forua);
    }

    /*See specific WAP browser documentation for information about support for this attribute
    * of the Meta tag.
    */
    public void setScheme(String scheme)
    {
        addAttribute("scheme",scheme);
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* An &lt;input&gt; tag with a text label next to it. See the Label class for more information↙
    on the arguments.
*/
public class LabeledInput extends Input
{
    String label;

    /**
    * @param label the text label to use with this Input tag
    */
    public LabeledInput(String name,String label) {
        super(name);
        this.label = label;
    }

    /**
    * @param label the text label to use with this Input tag
    */
    public LabeledInput(String name,String type,String format,String label)
    {
        super(name,type,format);
        this.label = label;
    }

    /**
    * @param label the text label to use with this Input tag
    */
    public LabeledInput(String name,String title,String type,String format,String value,     ↙
        String defaultValue,String label) {
        super(name,title,type,format,value,defaultValue);
        this.label = label;
    }

    /**
    * @param label the text label to use with this Input tag
    */
    public void setLabel(String label) {
        this.label = label;
    }

    public String getLabel() {
        return label;
    }

    public String render() {
        return label + super.render();
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;input&gt; element lets the user enter text which the device assigns to a specified ↙
    variable.
*/
public class Input extends WMLTag
{
    public final static String TYPE_TEXT = "text";
    public final static String TYPE_PASSWORD = "password";

    public final static String FORMAT_UCASE_ALPHA = "A";
    public final static String FORMAT_LCASE_ALPHA = "a";
    public final static String FORMAT_NUMERIC = "N";
    public final static String FORMAT_ANY_UCASE = "X";
    public final static String FORMAT_ANY_LCASE = "x";
    public final static String FORMAT_ANY_UCASE_CHANGEABLE = "M";
    public final static String FORMAT_ANY_LCASE_CHANGEABLE = "m";

    /**
    *@param name  The name of the variable in which the device stores the text entered by the↙
        user.
    */
    public Input(String name)
    {
        super("input",false);
        addAttribute("name",name);
    }

    /**
    *@param name  The name of the variable in which the device stores the text entered by the↙
        user.
    *@param title Specifies a brief label for the input item. Some devices use the label as a↙
        tooltip when displaying the input field. Others might use it as a label for a user   ↙
        interface mechanism that lets the user navigate to the item. For example, if a device↙
        cannot display all card content on one screen and ordered="true" (see Order          ↙
        options), the UP.Browser uses the title to identify this input item on a             ↙
        summary-level menu.
    *@param type (text | password) Specifies how the device should display text the user     ↙
        enters. Specifying type="text" causes the text to be visible. Specifying type        ↙
        ="password" causes the text to be masked (for example, replaced by "*" characters).  ↙
        Note that the password mode is not encrypted so you should not rely on it for        ↙
        securing critical data.
    *@param format Specifies a data format that the user entry must match (see Specifying a  ↙
        format mask below). If you omit this attribute, the device assumes *M (default       ↙
        uppercase first character followed by up to maxlength number of mixed case alphabetic↙
        and numeric characters).
    */
    public Input(String name,String type,String format) {
        this(name);
        addAttribute("type",type);
        setFormat(format);
    }

    /**
    *@param name  The name of the variable in which the device stores the text entered by the↙
        user.
    *@param title Specifies a brief label for the input item. Some devices use the label as a↙
        tooltip when displaying the input field. Others might use it as a label for a user   ↙
        interface mechanism that lets the user navigate to the item. For example, if a device↙
        cannot display all card content on one screen and ordered="true" (see Order          ↙
        options), the UP.Browser uses the title to identify this input item on a             ↙
        summary-level menu.
    *@param type (text | password) Specifies how the device should display text the user     ↙
        enters. Specifying type="text" causes the text to be visible. Specifying type        ↙
        ="password" causes the text to be masked (for example, replaced by "*" characters).  ↙
        Note that the password mode is not encrypted so you should not rely on it for        ↙
        securing critical data.
    *@param format Specifies a data format that the user entry must match (see Specifying a  ↙
        format mask below). If you omit this attribute, the device assumes *M (default ↙
        uppercase first character followed by up to maxlength number of mixed case alphabetic↙
        and numeric characters).
    *@param value  Specifies the value of the variable named in the name attribute. When the ↙
        element is displayed and the variable named in the name attribute is not set, the ↙
        name variable is assigned the value specified in the value attribute. If the name ↙
        variable already contains a value, the value attribute is ignored.
    */
    public Input (String name, String title, String type, String format, String value, String↙
        defaultValue)
    {
        this(name,type,format);
        addAttribute("title",title);
        addAttribute("value",value);
        addAttribute("default",defaultValue);
    }

    /**
    *@param type (text | password) Specifies how the device should display text the user ↙
        enters. Specifying type="text" causes the text to be visible. Specifying type ↙
        ="password" causes the text to be masked (for example, replaced by "*" characters). ↙
        Note that the password mode is not encrypted so you should not rely on it for ↙
        securing critical data.
    */
    public void setType (String type)
    {
        addAttribute("type",type);
    }

    /*
    *@param value  Specifies the value of the variable named in the name attribute. When the ↙
        element is displayed and the variable named in the name attribute is not set, the ↙
        name variable is assigned the value specified in the value attribute. If the name ↙
        variable already contains a value, the value attribute is ignored.
    */
    public void setValue (String value)
    {
        addAttribute("value",value);
    }

    /**
    * @param emptyOk (true | false) Specifies whether the user can leave the field blank. ↙
        Specifying
    * emptyok="true" indicates that the field is optional--if the user enters a value, ↙
        however, the
    * device applies any entry requirements you specify for the format attribute.
    */
    public void setEmptyOk (boolean emptyOk)
    {
        addAttribute("emptyok","" + emptyOk);
    }

    /*
    *@param format Specifies a data format that the user entry must match (see Specifying a ↙
        format mask below).
    *If you omit this attribute, the device assumes *M (default uppercase first character).
    */
    public void setFormat (String format)
    {
        addAttribute("format",format);
    }

    /*
    *@param format Specifies a data format that the user entry must match (see Specifying a ↙
        format mask below).
    *If you omit this attribute, the device assumes *M (default uppercase first character ↙
        followed by up to
    *maxlength number of mixed case alphabetic and numeric characters).
    */
    public void setFormat (String format,int max)
    {
        addAttribute("format",max + format);
    }

    /*Sets the 'size' attribute for the Input tag.
    *
    */
    public void setSize (int size)
    {
        addAttribute("size","" + size);
    }

    /**
    *@param maxlength  Specifies the maximum number of characters the user can enter. If you ↙
        do not specify the maxlength attribute, the UP.Browser imposes a limit of 256        ↙
        characters.
    */
    public void setMaximumLength(int maxLength)
    {
        addAttribute("maxLength","" + maxLength);
    }

    /*Sets the 'tabindex' attribute for the Input tag.
    *
    */
    public void setTabIndex (int tabIndex)
    {
        addAttribute("tabindex","" + tabIndex);
    }

    /*Returns the name of this Input tag.
    *
    */
    public String getInputName()
    {
        return getAttribute("name").getValue();
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;image&gt; element instructs the device to display an image within formatted text.  ↙
    Note that not all devices can display images.
*/
public class Image extends WMLTag
{
    public final static String ALIGN_TOP = "top";
    public final static String ALIGN_MIDDLE = "middle";
    public final static String ALIGN_BOTTOM = "bottom";

    /**
    *@param src The URL of the image to display. If you specify a valid icon for the localsrc↙
        attribute (see below), the device ignores this attribute.
    *@param alt Specifies the text to display if the device does not support images or cannot↙
        find the specified image.
    */
    public Image(String src,String alt) {
        super("img");
        if (src != null)
            addAttribute("src",src);
        else
            addAttribute("src","");

        if (alt != null)
            addAttribute("alt",alt);
        else
            addAttribute("alt"," ");
    }

    /**
    * @param icon A class representing a known icon. If the device cannot find the icon in    ↙
        ROM (Read-Only Memory), it attempts to retrieve it from the UP.Link Server. If you    ↙
        specify a valid icon (see Figure 2-6 for a list of icon names), the device ignores    ↙
        the src and alt attributes (see above) even though they are still required.
    * @param align alignment of image with text (ALIGN_TOP, ALIGN_MIDDLE, ALIGN_BOTTOM)
    * @param alt Specifies the text to display if the device does not support images or       ↙
        cannot find the specified image.
    */
    public Image(Icon icon,String align,String alt) {
        this(null,alt);
        addAttribute("localsrc",icon.getLocalSrc());
        addAttribute("align",align);
    }

    /**
    * @param height the desired scaled height of the image
    * @param width the desired scaled width of the image
    */
    public void setSize(int height,int width)
    {
        addAttribute("height","" + height);
        addAttribute("width","" + width);
    }

    /**
    * @param vspace the amount of space above and below the image
    * @param hspace the amount of space to the left and right of the image
    */
    public void setSpace(int vspace,int hspace)
    {
        addAttribute("hspace","" + hspace);
        addAttribute("vspace","" + vspace);
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* A utility class which represents the static strings used for indicating the names
* of the built-in icons which some WAP phones support for use with the Image tag.  Consult
* the documentation for specific WAP browsers for more information.
*/
public class Icon
{
    private String localSrc = null;

    /**
    * @param localSrc the local icon name to be used
    */
    public Icon (String localSrc)
    {
        this.localSrc = localSrc;
    }

    public String getLocalSrc() {
        return localSrc;
    }

    public final static String EXCLAMATION1 = "exclamation1";
    public final static String EXCLAMATION2 = "exclamation2";
    public final static String QUESTION1 = "question1";
    public final static String QUESTION2 = "question2";

    public final static String MAILBOX = "mailbox";

    public final static String MAGNIFY_GLASS = "magnifyglass";

    public final static String LOCK_KEY = "lockkey";

    public final static String INBOX = "inbox";
    public final static String OUTBOX = "outbox";

    public final static String FOLDER_CLOSED = "folder1";
    public final static String FOLDER_OPEN = "folder2";

    public final static String CLOCK = "clock";

    public final static String PUSHPIN = "pushpin";

    public final static String DOCUMENT1 = "document1";

    public final static String FLOPPY_DISK = "floppy1";

    public final static String CHECKMARK1 = "checkmark1";

    public final static String PHONE_OLD = "phone1";
    public final static String PHONE_HANDSET = "phone2";
    public final static String PHONE_MOBILE = "phone3";

    public final static String ENVELOPE1 = "envelope1";
    public final static String ENVELOPE2 = "envelope2";

    public final static String PAPERCLIP = "paperclip";

    public final static String PENCIL = "pencil";

    public final static String ROLOCARD = "rolocard";

    public final static String CALENDAR_MONTH = "calendar";
    public final static String CALENDAR_DAY = "day";

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;head&gt; element specifies information about the deck as a whole, including    ↙
    metadata and access control information.
*/
public class Head extends WMLTag
{
    public Head() {
        super("head");
    }

    /**
     * @param child a Tag to be added to this FieldSet.
     * @exception com.thinairapps.tag.InvalidTagException if the tag is not an instance or  ↙
        subclass of classes Access or Meta.
     */
    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Access || child instanceof Meta)
            super.addChild(child);
        else
            throw new InvalidTagException();
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;go&gt; element is a task element that instructs the device to open a specified URL ↙
    . If the URL specifies a particular card, the device displays that card. If the URL ↙
    specifies a deck, the device displays the first card in that deck.
*/
public class Go extends Task
{

    public final static String METHOD_GET = "get";
    public final static String METHOD_POST = "post";

    public final static String CHARSET_DEFAULT = "UTF-8";

    /***Constructs a standard Go Tag with the appropriate URL.
     *
     * @param href The URL to which this Task should link
     * @param closed a boolean indicating if there are to be children Tags
     */
    public Go(String href,boolean closed) {
        super("go",closed);
        addAttribute("href",href);
        addAttribute("sendreferer","true");
    }
    /***Constructs a standard Go Tag with the appropriate URL and link method.
     *
     * @param href The URL to which this Task should link
     * @param closed A boolean indicating if there are to be children Tags
     * @param method This should be either 'Go.METHOD_GET' or "Go.METHOD_POST"
     */
    public Go(String href,boolean closed,String method) {
        super("go",closed);
        addAttribute("href",href);
        addAttribute("method",method);
        addAttribute("sendreferer","true");
    }

    /***Constructs a Go Tag with the appropriate URL and link method.
     *
     * @param href The URL to which this Task should link
     * @param closed A boolean indicating if there are to be children Tags
     * @param sendReferrer A boolean indicating if there deck URL should be included in the ↙
        URL request
     * @param method This should be either 'Go.METHOD_GET' or "Go.METHOD_POST"
     * @paran acceptCharset Specifies the device encoding you application can handle in a ↙
        comma or space
     * delimited list, such as "UTF-8,US-ASCII,ISO,8859-1".
     */
    public Go(String href,boolean closed,boolean sendReferer,String method,String ↙
        acceptCharset) {
        this(href,closed);
        addAttribute("sendreferer","" + sendReferer);
        addAttribute("method",method);
        addAttribute("accept-charset",acceptCharset);
    }


    /***Adds a PostField to this Go task.
     */
    public void addPostField(PostField postfield) throws InvalidTagException {
        addChild(postfield);
    }

    /***Adds a PostField to this Go task with the specified value pre-set.
     */
    public void addPostField(String name,String value) throws InvalidTagException {
        addPostField(new PostField(name,value));
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;fieldset&gt; element allows you to group multiple text or input items within a card↙
    . Specifying
* one or more &lt;fieldset&gt; elements lets you control how the device presents card content↙
    in order to
* simplify user navigation.
*/
public class FieldSet extends WMLTag
{
    public FieldSet() {
        super("fieldset");
    }

    /**
    * @param title Specifies a brief label for the &lt;fieldset&gt; group. Some devices use  ↙
        the label as
    * a title when displaying the &lt;fieldset&gt; content. Others might use it as a label  ↙
        for a user
    * interface mechanism that lets the user navigate to the &lt;fieldset&gt; content.  ↙
        Consult individual
    * WAP browser documentation for more details.
    */
    public FieldSet(String title) {
        this();
        addAttribute("title",title);
    }

    /**
     * @param child a Tag to be added to this FieldSet.
     * @exception com.thinairapps.tag.InvalidTagException if the tag is not an instance or  ↙
        subclass of classes Text, FieldSet, Input, or Select.
     */
    public void addChild(WMLTag child) throws InvalidTagException {
        if (child instanceof Text || child instanceof FieldSet || child instanceof Input ||  ↙
            child instanceof Select)
            super.addChild(child);
        else
            throw new InvalidTagException();
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* The &lt;do&gt; element associates a task with an element within the user interface (for
    example, a
* function key, graphically-rendered button, or voice-activated command). When the user
    invokes the
* user interface mechanism, the device performs the associated &lt;do&gt; element task.
*/
public class Do extends WMLTag
{

    public final static String TYPE_ACCEPT = "accept";
    public final static String TYPE_DELETE = "delete";
    public final static String TYPE_HELP = "help";
    public final static String TYPE_OPTIONS = "options";
    public final static String TYPE_PREV = "prev";
    public final static String TYPE_RESET = "reset";
    public final static String TYPE_UNKNOWN = "";

    /**
    * @param type Identifies the generic user interface mechanism that triggers the specified
        &lt;do&gt;
    * element task (see descriptions below).
    * @param task Task tag to added as child
    */
    public Do (String type, Task task) throws InvalidTagException {
        super("do");
        addAttribute("type",type);
        addChild(task);
    }

    /**
    * @param type Identifies the generic user interface mechanism that triggers the specified
        &lt;do&gt;
    * element task (see descriptions below).
    * @param task Task tag to added as child
    * @param label A label that identifies the task with the user interface mechanism. For
        example, if
    * you bind a task to the ACCEPT key, the device displays this value as the function key
        label. If
    * you do not specify the label attribute, the device uses the word "OK" as the default
        ACCEPT key
    * label. To ensure compatibility on a wide range of devices, label should be a maximum of
        five
    * characters. Devices ignore the label attribute if they do not support dynamic labelling
        .
    * @param name  Specifies a name for the &lt;do&gt; element. If a card-level &lt;do&gt;
        element (i.e.
    * defined within a &lt;card&gt; element) has the same name as a deck-level &lt;do&gt;
        element (i.e.
    * defined within a &lt;template&gt; element), the card-level binding overrides the
        deck-level binding.
    * @param optional Specifies whether the device can ignore this element.
   */
    public Do (String type, Task task, String label, String name, boolean optional)  throws
        InvalidTagException {
        this(type,task);
        addAttribute("label",label);
        addAttribute("name",name);
        addAttribute("optional","" + optional);
    }

    /**
    * @param type Identifies the generic user interface mechanism that triggers the specified
        &lt;do&gt;
    * element task.  This should be one of the type constants defined in this class.
```

```
     * @param task Task tag to added as child
     * @param label A label that identifies the task with the user interface mechanism. For  ↙
         example, if
     * you bind a task to the ACCEPT key, the device displays this value as the function key ↙
         label. If
     * you do not specify the label attribute, the device uses the word "OK" as the default  ↙
         ACCEPT key
     * label. To ensure compatibility on a wide range of devices, label should be a maximum of↙
         five
     * characters. Devices ignore the label attribute if they do not support dynamic labelling↙
         .
     * @param name  Specifies a name for the &lt;do&gt; element. If a card-level &lt;do&gt;  ↙
         element (i.e.
     * defined within a &lt;card&gt; element) has the same name as a deck-level &lt;do&gt;   ↙
         element (i.e.
     * defined within a &lt;template&gt; element), the card-level binding overrides the      ↙
         deck-level binding.
     * @param optional Specifies whether the device can ignore this element.
    */
     public Do (String type, Task task, String label, boolean optional)  throws              ↙
         InvalidTagException {
         this(type,task);
         addAttribute("label",label);
         addAttribute("optional","false");
     }
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
* A widget which extends Card and providers basic functionality for displaying a message and ↙
    a link
*/
public class DisplayCard extends Card
{

    public DisplayCard()
    {
        super();
    }

    /**Constructs a new Card with the given name ('id') attribute.
     *
     * @param id specifies a name for the card
     */
    public DisplayCard(String id)
    {
        super(id);
    }

    /**Constructs a new Card with the given name ('id') and label ('title') attribute.
     *
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     */
    public DisplayCard(String id,String title)
    {
        super(id,title);
    }

    /**Constructs a new Card with the given name ('id'), label ('title') and 'newcontext'    ↙
        attributes.
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     * @param newContext specifies whether the device should initialize the context whenever ↙
        the user naviages to the card through a &lt;go/&gt; task
     */
    public DisplayCard(String id,String title,boolean newContext)
    {
        super(id,title,newContext);
    }

    /**Constructs a new Card with the given name ('id'), label ('title'), 'newcontext',
     * and 'ordered' attributes.
     *
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     * @param newContext specifies whether the device should initialize the context whenever ↙
        the user naviages to the card through a &lt;go/&gt; task
     * @param ordered specifies the organization of card content
     */
    public DisplayCard(String id,String title,boolean newContext,boolean ordered)
    {
        super(id,title,newContext,ordered);
    }

    /**
    * @param text the text you wish to display
    * @param align the paragraph alignment to use for the next (Paragraph.LEFT, Paragraph.   ↙
        CENTER, Paragraph.RIGHT)
    */
    public void buildCard (String text, String align)
    {
            Paragraph p = new Paragraph(align,Paragraph.MODE_WRAP);
            p.addChild(new Text(text));
            addParagraph(p);
    }


    /**
    * @param text the text you wish to display
    * @param align the paragraph alignment to use for the next (Paragraph.LEFT, Paragraph. ↙
        CENTER, Paragraph.RIGHT)
    * @param href the url to use with the "Ok" link on the Card
    */
    public void buildCard (String text, String align, String href)
    {
        try {
            addChild(new Do(Do.TYPE_ACCEPT,new Go(href,false)));
            Paragraph p = new Paragraph(align,Paragraph.MODE_WRAP);
            p.addChild(new Text(text));
            addParagraph(p);
        }
        catch(InvalidTagException e) {
            e.printStackTrace();
        }
    }

    /**
    * @param text the text you wish to display
    * @param align the paragraph alignment to use for the next (Paragraph.LEFT, Paragraph. ↙
        CENTER, Paragraph.RIGHT)
    * @param href the url to use with the "Ok" link on the Card
    * @param seconds the amount of time before the href link should be automatically loaded ↙
        (uses setOnTimer())
    */
    public void buildCard (String text, String align, String href, int seconds)
    {
        Paragraph p = new Paragraph(align,Paragraph.MODE_WRAP);
        Timer timer = new Timer(seconds);
        setOnTimer(href);
        p.addChild(timer);
        p.addChild(new Text(text));
        addParagraph(p);
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* A utility class solely used to group sets of WMLTags together for rendering.  This is
    public
* class as an implementaion detail.  Do not use this class in your applications.
*/
public class Container extends WMLTag
{
    public Container() {
        super("",false);
    }

    public String render() {
        return renderChildren();
    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.*;

/**
* A non-rendered portion of the page used to contain any extraneous information desired by
    the developer
*/
public class Comment extends WMLTag {

    /**Constructs a Comment with the given String.
    *
    * @param text the comment to be added to the WML Deck
    */
    public Comment(String text) {
        super("!--  " + text + "  --",false);
    }

}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.InvalidTagException;

/**
 * A WML deck consists of one or more &lt;card&gt; elements, each of which specifies
 * a single interaction between the user and the device.
 */
public class Card extends WMLTag
{
    public Card() {
        super("card");
    }

    /**Constructs a new Card with the given name (id).
     *
     * @param id specifies a name for the card
     */
    public Card(String id) {
        this();
        addAttribute("id",id);
    }

    /**Constructs a new Card with the given name ('id') and label ('title') attributes.
     *
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     */
    public Card(String id,String title) {
        this(id);
        addAttribute("title",title);
    }

    /**Constructs a new Card with the given name (id), label (title) and 'newcontext'
        attributes.
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     * @param newContext specifies whether the device should initialize the context whenever
        the user naviages to the card through a &lt;go/&gt; task
     */
    public Card(String id,String title,boolean newContext) {
        this(id,title);
        addAttribute("newcontext","" + newContext);
    }

    /**Constructs a new Card with the given name ('id'), label ('title'), 'newcontext'
        attributes,
     * and 'ordered' attributes.
     *
     * @param id specifies a name for the card
     * @param title specifies a brief label for the card
     * @param newContext specifies whether the device should initialize the context whenever
        the user naviages to the card through a &lt;go/&gt; task
     * @param ordered specifies the organization of card content
     */
    public Card(String id,String title,boolean newContext,boolean ordered) {
        this(id,title,newContext);
        setOrdered(ordered);
    }

    /**This attribute has different effects on different browsers.  Consult your WAP browser
     * documentation for information about how ordered and unordered Cards are rendered.
     *
     * @param ordered specifies the organization of card content.
     */
    public void setOrdered(boolean ordered) {
        addAttribute("ordered","" + ordered);
    }
    /**
     * @param url specifies the URL to open if the user navigates to this card through a &lt;↙
        go&gt; task
     */
    public void setOnEnterForward(String url) {
        addAttribute("onenterforward",url);
    }

    /**
     * @param url specifies the URL to open if the user navigates to this card through a &lt;↙
        prev&gt; task
     */
    public void setOnEnterBackward(String url) {
        addAttribute("onenterbackward",url);
    }

    /**
     * @param url specifies the URL to open if a specified &lt;timer&gt; element expires
     */
    public void setOnTimer(String url) {
        addAttribute("ontimer",url);
    }

    /**Adds a Paragraph to this Card.
     * @param p Paragraph tag to be added
     */

        public void addParagraph(Paragraph p) {
          try { addChild(p); }
          catch(InvalidTagException e) {
            e.printStackTrace();
          }
        }


    /**
     * @param child WMLTag to be added
     */
    public void addChild(WMLTag child) throws InvalidTagException {

        if(!(child instanceof OnEvent || child instanceof Timer || child instanceof Do ||        ↙
            child instanceof Anchor || child instanceof FieldSet || child instanceof Image      ↙
            || child instanceof Input || child instanceof Select || child instanceof            ↙
            Paragraph || child instanceof Container))
            throw new InvalidTagException("invalid child tag");
        else
            super.addChild(child);
    }

}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;br/&gt; element specifies a line break. For example, it causes the device to ↙
    display the subsequent text or image on a new line.
*/
public class Break extends WMLTag
{
    public Break()
    {
        super ("br", false);
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;b&gt; element specifies bold text.
*/
public class Bold extends WMLTag
{
    /**Constructs an empty Bold tag.
    *
    */
    public Bold()
    {
        super("b",true);
    }

    /**Constructs a Bold tag with the given WMLTag as a child.
    *
    */
    public Bold (WMLTag tag)
    {
        this();
        addChild(tag);
    }

    /**Constructs a Bold tag with the given Text tag as a child.
    *
    */
    public Bold (String text)
    {
        this (new Text(text));
    }
}
```

```
package com.thinairapps.tag.wml;

import com.thinairapps.tag.Attribute;
import com.thinairapps.tag.InvalidTagException;
import java.util.Enumeration;

/**
* anchor element: The &lt;anchor&gt; element anchors a task to a string of formatted text, ↙
    often called a link. You can specify a link within any formatted text or image. When a ↙
    user selects the link and presses ACCEPT, the device executes the task.
*/
public class Anchor extends WMLTag
{
    /**
    * @param task represents the action to perform when the user activates the link and text ↙
        is the text the device will display to represent the link. You must anchor one of the↙
        following task elements to a link: &lt;go&gt;, &lt;prev&gt;, &lt;refresh&gt;, &lt; ↙
        spawn&gt;, &lt;exit&gt;, &lt;throw&gt;
    *  @param text Devices typically set this text off from surrounding text, for instance, ↙
        by enclosing it in square brackets (see example) or underlining it if the device can ↙
        display bitmap images.
    */
    public Anchor (Task task, Text text)
    {
        super("anchor");
        addChild(task);
        addChild(text);
    }

     /**
    * @param task represents the action to perform when the user activates the link and text ↙
        is the text the device will display to represent the link. You must anchor one of the↙
        following task elements to a link: &lt;go&gt;, &lt;prev&gt;, &lt;refresh&gt;, &lt; ↙
        spawn&gt;, &lt;exit&gt;, &lt;throw&gt;
    * @param text Devices typically set this text off from surrounding text, for instance, by↙
        enclosing it in square brackets (see example) or underlining it if the device can ↙
        display bitmap images.
    * @param title A label that identifies the link. If you do not specify the title ↙
        attribute, the device uses the word "Link" as the default label.
    */
    public Anchor(Task task,Text text,String title) {
        this(task,text);
        addAttribute("title",title);
    }

    /**
    * @param href url link for action
    * @param title title displayed on button when selected
    * @param generally child text to display as link
    */
    public Anchor(String href,String title,WMLTag child) {
        super("a");
        addAttribute("href",href);
        if (title != null)
         addAttribute("title",title);
        addChild(child);
    }

    /**
     * Used to add a child tag to this Anchor object.
     *
     * @param child A WMLTag object to add as a child of this tag
     */
    public void addChild(WMLTag tag) {
      try { super.addChild(tag); }
      catch(InvalidTagException e) {
        e.printStackTrace();
      }
    }
    protected String renderOpenTag() {
        StringBuffer output = new StringBuffer();

        //render self open
        output.append("<");
        output.append(name);

        Enumeration eAttribs = attributes.elements();

        while(eAttribs.hasMoreElements())
            output.append(" " + ((Attribute)eAttribs.nextElement()).render());

        output.append(">");

        return output.toString();
    }

    protected String renderChildren() {

        StringBuffer output = new StringBuffer();

        Enumeration eChildren = children.elements();

        while(eChildren.hasMoreElements())
            output.append(((WMLTag)eChildren.nextElement()).render());

        return output.toString();
    }

    protected String renderCloseTag() {
        if (closingTag)
            return "</" + name + ">";
        else
            return "";
    }
}
```

```
package com.thinairapps.tag.wml;

/**
* The &lt;access&gt; element specifies access control information for a WML deck.
* You must specify this element within the deck header along with any meta
* information for the deck (for more information, see &lt;head&gt; element and
* &lt;meta&gt; element). Each deck can have only one &lt;access&gt; element. All
* WML decks are public by default.
*/
public class Access extends WMLTag
{
    public Access() {
        super("access",false);
    }

    /**
    * @param domain The URL domain of other decks that can access cards in the
    * deck. The default value is the domain of the current deck.
    */
    public void setDomain(String domain) {
        addAttribute("domain",domain);
    }

    /**
    * @param path The URL root of other decks that can access cards in the deck.
    * The default value is "/" (the root path of the current deck) which lets any
    * deck within the specified domain access this deck.
    */
    public void setPath(String path) {
        addAttribute("path",path);
    }
}
```

```
package com.thinairapps.tag.wml.goWebRim;

import com.thinairapps.tag.wml.*;

/**
* The &lt;input&gt; element lets the user enter text which the device assigns to a specified ↙
    variable.
*/
public class TextArea extends WMLTag
{
    public final static String TYPE_TEXT = "text";
    public final static String TYPE_PASSWORD = "password";

    public final static String FORMAT_UCASE_ALPHA = "A";
    public final static String FORMAT_LCASE_ALPHA = "a";
    public final static String FORMAT_NUMERIC = "N";
    public final static String FORMAT_ANY_UCASE = "X";
    public final static String FORMAT_ANY_LCASE = "x";
    public final static String FORMAT_ANY_UCASE_CHANGEABLE = "M";
    public final static String FORMAT_ANY_LCASE_CHANGEABLE = "m";

    public TextArea(String name)
    {
        super("textarea",true);
        addAttribute("name",name);
    }

    /**
     * @param name the unique id of this element
     * @param rows the number of rows
     * @param cols the number of columns
     */
    public TextArea(String name,int rows,int cols) {
        super("textarea",true);
        addAttribute("name",name);

        addAttribute("rows","" + rows);
        addAttribute("cols","" + cols);
    }

    /**
     * @param text set the text value for this element
     */
    public void setValue(String text) {
        getChildren().removeAllElements();
        getChildren().addElement(new Text(text));
    }
}
```

```
package com.thinairapps.tag.wml.goWebRim;

/**
* An &lt;input&gt; tag with a text label next to it. See the Label class for more information↙
    on the arguments.
* This class is to be used with the Go Web browser.
*/
public class LabeledTextArea extends TextArea
{
    String label;

    /**
    * @param label the text label to use with this TextArea tag
    */
    public LabeledTextArea(String name,String label) {
        super(name);
        this.label = label;
    }

    /**
    * @param label the text label to use with this TextArea tag
    */
    public LabeledTextArea(String name,int rows,int cols,String label)
    {
        super(name,rows,cols);
        this.label = label;
    }

    /**
    * @param label the text label to use with this Input tag
    */
    public void setLabel(String label) {
        this.label = label;
    }

    public String getLabel() {
        return label;
    }

    public String render() {
        return label + super.render();
    }
```

```
package com.thinairapps.tag;

/**
 * The Attribute class is used to store all name/value pairs within a Tag object.  This
    represents
 * a standard XML attribute
 */
public class Attribute
{
    String name;
    String value;
    boolean useQuotes;

    private static final String NO_NAME = "_NONE";

    /**
     * @param name The attribute name
     * @param value The attribute value
     * @param useQuotes determines whether the value should be surrounded by quotation marks
     */
    public Attribute (String name, String value, boolean useQuotes)
    {
        this.name = name;
        this.value = value;
        this.useQuotes = useQuotes;
    }

    /**
     * @param name The attribute name
     * @param value The attribute value
     */
    public Attribute(String name,String value)
    {
        this(name,value,true);
    }

    /**
     * @param value A standalone value to be used within a tag.  The name defaults to
     * '_NONE'.
     */
    public Attribute (String value)
    {
        this(NO_NAME,value,false);
    }

    /**
     * @return the attribute name
     */
    public String getName()
    {
        return name;
    }

    /**
     * @return the attribute value
     */
    public String getValue() {
        return value;
    }

    /**
     * @return a string object representing the fully rendered text for this attribute
     */
    public String render()
    {
        if (name.equals(NO_NAME)) {
            return value;
        }
        else {
            if (useQuotes)
                return name + "=\"" + value + "\"";
            else
                return name + "=" + value;
        }
    }
}
```

```java
package com.thinairapps.tag;

/**
 * This runtime expection is thrown when addChild() is called, and passed a tag which
 * the class doesn't support as a child tag
 */
public class InvalidTagException extends RuntimeException
{
    /**
     * @param message text to display for this exception
     */
    public InvalidTagException (String message)
    {
        super(message);
    }

    /**
     *
     */
    public InvalidTagException()
    {
        super("invalid tag used");
    }
}
```

```
package com.thinairapps.tag;

import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;

/**
 * This class is the basic Tag class used to implement all
 * other Tag elements. It can be extended to implement a new markup
 * language. A Tag can have any number of Attributes, be standalone
 * or paired, and have a hierarchy of children Tags.
 */
public class Tag
{

    public String name;
    public Hashtable attributes;
    public Vector children;
    public boolean closingTag;

    /**
     * A constructor which creates a tag with the given name, and which may either be ↙
        standalone or a pair.
     *
     * @param name the text to use for this tag (i.e. 'body' would be the value for &lt;body&↙
        gt;)
     * @param closingTag indicates whether this tag has a paired closing tag or is standalone
    */
    public Tag (String name, boolean closingTag)
    {
        this.name = name;
        attributes = new Hashtable();
        children = new Vector();
        this.closingTag = closingTag;
    }

    /**
     * A constructor which creates a paired tag set with the given name.
     *
     * @param name the text to use for this tag (i.e. 'body' would be the value for &lt;body&↙
        gt;)
     */
    public Tag (String name)
    {
        this(name,true);
    }

    /**
     * @return name of this tag
     */
    public String getName()
    {
        return name;
    }

    /**
     * @return Vector of all child tag of this tag. Note that the children themselves may be ↙
        parent nodes.
     */
    public Vector getChildren()
    {
        return children;
    }

    /**
     * @return A Hashtable of all attributes for this tag
     */
    public Hashtable getAttributes()
    {
        return attributes;
    }

    /**
     * @return indicated whether this Tag has a closed pair or is open and standalone
     */
    public boolean isClosedTag()
    {
        return this.closingTag;
    }

    /**
     * Sets the name of this tag, after it has been constructed.
     *
     * @param name the text to use for this tag (i.e. 'body' would be the value for &lt;body&↙
        gt;)
     */
    public void setName (String name)
    {
        this.name = name;
    }

    /**
     * Used to add an attribute object to this tag
     *
     * @param attrib The attribute object to add to this tag
     */
    public void addAttribute (Attribute attrib)
    {
        attributes.put(attrib.getName(),attrib);
    }

    /**
     * Used to add an attribute, in the form of a name/value pair, to this tag
     *
     * @param name the attribute name
     * @param value the attribute value
     */
    public void addAttribute (String name, String value)
    {
        addAttribute(new Attribute(name,value));
    }

    /**
     * Used to retrieve an existing attribute for this tag object
     *
     * @param name the name of the attribute object to retrieve
     * @return an attribute object corresponding to the name value passed; will be NULL if   ↙
        attribute doesn't exist
     */
    public Attribute getAttribute (String name)
    {
        return (Attribute)attributes.get(name);
    }

    /**
     * Used to add a child tag to this tag object. Intended for use only
     * with paired or "closed" tags, but won't throw an exception either
     * way. Tag never throws an InvalidTagException, but classes which
     * extend Tag may to enforce restrictions of certain markup languages.
     *
     * @param child A Tag object to add as a child of this tag
     */
    public void addChild (Tag child) throws InvalidTagException
    {
        children.addElement(child);
    }

    /**
```

```
     * Clears all child tags from this tag.
     */
    public void clearChildren()
    {
        children.removeAllElements();
    }

    /**
     * This method is currently not implemented. The goal of it is however to
     * allow a markup language text to be passed to it, and have a complete object
     * hierarchy be created- similar to an XML DOM Parser.
     */
    public void parse (String fullTagText)
    {

    }

    /**
    *
    * This method is intended to be used as a way of creating a tag less version
    * of a tag hierarchy. For instance, you may create a complete HTML Tag hierarchy
    * of a webpage, either directly or through parsing, and then want just the text
    * content from that page for display on a WAP Phone or Pager.
    *
    * @return only "intra-tag" text content for itself, and all subtags.
    */
    public String getTextOnly()
    {

        Enumeration eChildren = children.elements();

        StringBuffer text = new StringBuffer();

        while(eChildren.hasMoreElements())
            text.append(((Tag)eChildren.nextElement()).getTextOnly() + " ");

        return text.toString();

    }

    /**
     * Renders opening tag, if a paired set, or the only tag if otherwise. All
     * attributes are also rendered as part of this Tag.
     *
     * @return formatted markup content of opening tag
     */
    protected String renderOpenTag()
    {
        StringBuffer output = new StringBuffer();

        //render self open
        output.append("<");
        output.append(name);

        Enumeration eAttribs = attributes.elements();

        while(eAttribs.hasMoreElements())
            output.append(" " + ((Attribute)eAttribs.nextElement()).render());

        output.append(">\n");

        return output.toString();
    }

    /**
     * Loops through all children Tag objects and calls their render() methods,
     * which, if parent tags themselves, would cause them to render() their children.
     *
     * @return formatted markup text of all child Tags of this Tag
     */
    protected String renderChildren()
    {

        StringBuffer output = new StringBuffer();

        Enumeration eChildren = children.elements();

        while(eChildren.hasMoreElements())
            output.append(((Tag)eChildren.nextElement()).render());

        return output.toString();
    }

    /**
     * Renders closed tag, if paired set, otherwise returns empty String
     *
     * @return formatted markup text of closing tag, or emtyp String if no closingTag
     */
    protected String renderCloseTag()
    {
        if (closingTag)
            return "</" + name + ">";
        else
            return "";
    }

    /**
     * Returns the markup String for this Tag and all of its children.
     *
     * @returns formatted markup text of this tag, all attributes, and all child tags.
     */
    public String render()
    {
        StringBuffer output = new StringBuffer();
        output.append(renderOpenTag());
        output.append(renderChildren());
        output.append(renderCloseTag());
        return output.toString();
    }

}
```

```
package com.thinairapps.tag;

/**
 * @(#)TagDocument.java
 * This abstract class functions as the container for all tag-based documents.  It is
   extended by
 * classes like WMLTagDocument HDMLTagDocument, and HTMLTagDocument to implement the
   specifics of
 * each document type.
 * <p>
 * In general, you will only want to use this class directly when implementing connectors
   that render
 * more than one type of markup.  This class gives you a means of having a reference to a
   TagDocument without
 * concerning yourself the type of browser to which the document is to be rendered.  When
   rendering for only one
 * browser, you can just use HTMLTagDocument, WMLTagDocument, etc. directly.
 * </p>
 */
public abstract class TagDocument
{
    private String location;
    private Tag root;
    private String contentType;
    private String docType;

    /**
     * Primary constructor used to build a specific type of TagDocument hierarcy.
     *
     * @param rootTag the name of the root document tag. (i.e. "HTML" for a webpage, or
         "WML" for a WAP Deck)
     * @param contentType the MIME content-type associated with the markup language being
         generated.
     * @param closed indicates whether the root tag is opened, or closed/paired.
     */
    public TagDocument (String rootTag, String contentType, boolean closed)
    {
        root = new Tag (rootTag, closed);
        this.contentType = contentType;
    }

    /**
     * Primary constructor used to build a specific type of TagDocument hierarcy.
     * Assumes that the root tag is a paired or closed.
     *
     * @param rootTag the name of the root document tag. (i.e. "HTML" for a webpage, or
         "WML" for a WAP Deck)
     * @param contentType the MIME content-type associated with the markup language being
         generated.
     */
    public TagDocument (String rootTag, String contentType)
    {
        this(rootTag,contentType,true);
    }

    /**
     * Sets the root node to a new Tag object.
     *
     * @param root The Tag object to use as the root tag for this document.
     */
    public void setRoot (Tag root)
    {
        this.root = root;
    }

    /**
     * Gets the root Tag object for this document.
     *
     * @return The root Tag object for this document.
     */
    public Tag getRoot()
    {
        return root;
    }

    /**
     * Adds a child Tag to the root Tag of this document.
     *
     * @param child A Tag object to add as a child of the root Tag.
     */
    public void addChild (Tag child) throws InvalidTagException
    {
        getRoot().addChild(child);
    }

    /**
     * Returns the entire rendered document text, suitable for display
     * in a browser which supports the MIME type specified by the document's
     * content-type.
     *
     * @param formatted markup language as String
     */
    public String render()
    {
        return root.render();
    }

    /**
     * The Internet MIME content-type which this document supports.
     *
     * @return an official internet mime-type such as text/html, or text/vnd.wap.wml
     */
    public String getContentType()
    {
        return contentType;
    }
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * Used to define intra or inter page hyperlinking action.
 */
public class Anchor extends HTMLTag
{
    /**
     * @param name anchor name (used for intra page linking)
     */
    public Anchor(String name)
    {
        super("a",true);
        if (name != null && name.length() > 0)
                  addAttribute("name",name);
    }

    /**
     *
     * @param name anchor name (used for intra page linking)
     * @param child a child node to wrap within this tag
     */
    public Anchor (String name,HTMLTag child)
    {
        this(name);
        addChild(child);
    }

    /**
     *
     * @param name anchor name (used for intra page linking)
     * @param href the URL or #tagname which this anchor should link to
     */
    public Anchor(String name, String href)
    {
        this(name);
        addAttribute("href",href);
    }

    /**
     *
     * @param name anchor name (used for intra page linking)
    * @param href the URL or #tagname which this anchor should link to
     * @param child a child node to wrap within this tag
     */
    public Anchor(String name,String href,HTMLTag child)
    {
        this(name,child);
        addAttribute("href",href);
    }

    /**
     * @param name anchor name (used for intra page linking)
    * @param href the URL or #tagname which this anchor should link to
    * @param target the name of target window or frame which to target the hyperlink action
        to
     * @param child a child node to wrap within this tag
     */
    public Anchor(String name,String href,String target,HTMLTag child)
    {
        this(name,child);
        addAttribute("href",href);
        addAttribute("target",target);
    }

    /**
     * @param action set a client-side scripting event
     */
    public void setOnMouseOver(String action)
    {
        addAttribute("onMouseOver",action);
    }

    /**
     * @param action set a client-side scripting event
     */
    public void setOnMouseOut(String action) {
        addAttribute("onMouseOut",action);
    }

    /**
     * @param action set a client-side scripting event
     */
    public void setOnClick(String action) {
        addAttribute("onClick",action);
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * The main content tag for an html page
 * @(#)Body.java
 *
 */
public class Body extends HTMLTag
{
    Tag header;
    Tag footer;

    /**
     * Basic constructor
     */
    public Body()
    {
        super("body",true);
    }

    /**
     * @param bgColor hex or name value for page background color
     * @param fgColor hex or name value for text font color
     * @param linkColor hex or name value for link font color
     */
    public Body(String bgColor,String textColor,String linkColor) {
        this();
        addAttribute("bgColor",bgColor);
        addAttribute("text",textColor);
        addAttribute("link",linkColor);
        addAttribute("alink",linkColor);
        addAttribute("vlink",linkColor);
    }


    /**
     * @param background url to a background image file
     * @param bgColor hex or name value for page background color
     * @param fgColor hex or name value for text font color
     * @param linkColor hex or name value for link font color
     */
    public Body(String background,String bgColor,String fgColor,String linkColor) {
        this(bgColor,fgColor,linkColor);
        addAttribute("background",background);
    }

    /**
     * @param p paragraph element to add as child
     */
    public void addParagraph(Paragraph p) {
      try { addChild(p); }
      catch(InvalidTagException e) {}
    }


    /**
     * @param action scripting action to perform when page is loaded
     */
    public void setOnLoad(String action) {
        addAttribute("onLoad",action);
    }

    /**
     * @param header HTMLTag to display at the top of each page
     */
    public void setHeader(HTMLTag header)
    {
        this.header = header;
    }


    /**
     * @param footer HTMLTag to display at the bottom of each page
     */
    public void setFooter(HTMLTag footer)
    {
        this.footer = footer;
    }

    public String render()
    {
        StringBuffer output = new StringBuffer();

        output.append(renderOpenTag());

                if (header != null)
          output.append(header.render());

        output.append(renderChildren());

                if (footer != null)
          output.append(footer.render());

        output.append(renderCloseTag());

        return output.toString();
    }
```

}

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.Tag;

/**
 * A text formatting node that indicates the text that should be displayed in bold.
 */
public class Bold extends HTMLTag
{
    /**
     * Basic Constructor
     */
    public Bold ()
    {
        super("b");
    }

    /**
     * @param text content that should be displayed in BOLD
     */
    public Bold (String text)
    {
        this();
        addChild(new Text(text));
    }

    /**
     * @param tag HTMLTag that should be added as default child to this tag
     */
    public Bold (HTMLTag tag)
    {
        this();
        addChild(tag);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * &lt;br&gt;- inserts a line break into the content
 */
public class Break extends HTMLTag
{
    public Break() {
        super("br",false);
    }

        public String render() {
          return "<BR>";
        }
}
```

```
package com.thinairapps.tag.html;

/**
 * A form element that can be used to trigger actions
 */
public class Button extends Input
{
    /**
     * @param name the unique identifier for this button
     */
    public Button(String name){
        super("button",name);
    }

    /**
     * @param name the unique identifier for this button
     * @param value the displayed text on the button
     */
    public Button(String name,String value)
    {
        super("button",name,value);
    }

    /**
     * @param action scriptable action caused when button is clicked
     */
    public void setOnClick(String action)
    {
        addAttribute("onClick",action);
    }
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * All children tags of this tag will be centered on the page
 */
public class Center extends HTMLTag
{
    public Center()
    {
        super("center",true);
    }

    /**
     * Construct with a default child tag
     *
     * @param child the default child tag to be centered
     */
    public Center (HTMLTag child)
    {
        super("center",true);
        addChild(child);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A form element indicating a boolean state
 */
public class CheckBox extends Input
{
    public final static int LABEL_BEFORE = 0;
    public final static int LABEL_AFTER = 1;

    Tag label;
    int orientation;

    /**
     * @param name the unique
     * @param value the default value to submit (if checked is true)
     * @param checked indicates state of checkbox
     */
    public CheckBox (String name,String value,boolean checked)
    {
        super("checkbox",name,value);

        if(checked)
            addAttribute(new Attribute("checked"));
    }

    /**
     * @param name the unique
     * @param value the default value to submit (if checked is true)
     * @param checked indicates state of checkbox
     * @param label a tag class to use as a label for this element
     * @param orientation a constant indicating position of label (LABEL_BEFORE||LABEL_AFTER)
     */
    public CheckBox(String name,String value,boolean checked,HTMLTag label,int orientation) {
        this(name,value,checked);
        setLabel(label,orientation);
    }


    /**
    * @param label a tag class to use as a label for this element
     * @param orientation a constant indicating position of label (LABEL_BEFORE||LABEL_AFTER)
     */
    public void setLabel(Tag label,int orientation)
    {
        this.label = label;
        this.orientation = orientation;
    }

    public String render() {

        StringBuffer output = new StringBuffer();

        if (label != null && orientation == LABEL_BEFORE)
            output.append(label.render());

        output.append(super.render());

        if (label != null && orientation == LABEL_AFTER)
            output.append(label.render());

        return output.toString();

    }

}
```

```java
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * Insert a hidden text comment into the page
 */
public class Comment extends HTMLTag
{
    /**
     * @param text message to put in comment
     */
    public Comment(String text) {
        super("!--  " + text + "  --",false);

    }

}
```

```
package com.thinairapps.tag.html;

/**
 * A link to a cascading stylesheet document- should be put in page header
 */
public class CSSLink extends Link
{
    /**
     * @param href url link to CSS document
     */
    public CSSLink (String href)
    {
        super ("stylesheet", "text/css", href);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;

/**
 * A basic widget for creating a body tag that displays some text. You must
 * call the Construcor and buildPage() for the tag hierarchy to be created.
 */
public class DisplayBody extends Body
{

    public DisplayBody()
    {
        super();
    }

    /**
     * @param text message to be displayed
     * @param align alignment (Paragraph.CENTER, etc.) to use on text
     */
    public void buildPage (String text, String align)
    {

            Paragraph p = new Paragraph(align);
            p.addChild(new Text(text));
            addParagraph(p);
    }

    /**
     * @param text message to be displayed
     * @param align alignment (Paragraph.CENTER, etc.) to use on text
     * @param href url link to insert into page via confirmation button (i.e. "Ok")
     */
    public void buildPage (String text, String align, String href)
    {
        try {
            Paragraph p = new Paragraph(align);
            p.addChild(new Text(text));
            addParagraph(p);
            addChild(new Break());
            addChild(new Anchor("",href,new Text("Ok")));

        }
        catch(InvalidTagException e) {
            e.printStackTrace();
        }
    }


}
```

```java
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A tag indicated text styling parameters
 */
public class Font extends HTMLTag
{
    /**
     * @param face the font face to apply to all children text tags
     */
    public Font(String face)
    {
        super("font",true);
        addAttribute("face",face);
    }

    /**
     * @param face the font face to apply to all children text tags (i.e. Arial, Helvetica)
     * @param size the font size to apply to all children text tags
     */
    public Font(String face,int size)
    {
        this(face);
        addAttribute("size","" + size);
    }

    /**
     * @param face the font face to apply to all children text tags (i.e. Arial, Helvetica)
     * @param size the font size to apply to all children text tags
     * @param color a hex or named color value to apply to all children text tags
     */
    public Font(String face,int size,String color)
    {
        this(face,size);
        addAttribute("color","#" + color);
    }
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A Form element for gathering user input and submitting it to an HTTP server
 */
public class Form extends HTMLTag
{

    /**
     * @param name the unique form id
     */
    public Form (String name)
    {
        super("form",true);
                addAttribute("name",name);
    }

    /**
     * @param name the unique form id
     * @param action the URL which to post the form data to
     * @param method the HTTP method which to submit the data with (POST, GET, PUT)
     */
    public Form(String name,String action,String method) {
        this(name);
        addAttribute("action",action);
        addAttribute("method",method);
    }

    /**
     * @param elem add a FormElement subclass as a child to this form
     */
        public void addFormElement(FormElement elem) {
          try {
              addChild(elem);
             // if (!(elem instanceof HiddenInput))
               // addChild(new Break());
          }
          catch(InvalidTagException e) {}

        }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * An element used as a base class for all form input elements
 */
public class FormElement extends HTMLTag
{
    /**
     * @param tagName the name of the form element tag ("input","button")
     * @param name the unique id of this element
     * @param closedTag indicates if this tag is standalone or has a closing pair
     */
    public FormElement(String tagName,String name,boolean closedTag)
    {
        super(tagName,closedTag);
        addAttribute("name",name);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * Tag to create header section within an HTML page
 */
public class Head extends HTMLTag
{
    public Head() {
        super("head",true);
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A Form element used for setting hidden variable values within a form
 */
public class HiddenInput extends Input
{
    /**
     * @param name the unique id for this element
     * @param value the value to send for this variable
     */
    public HiddenInput (String name,String value)
    {
        super("hidden",name,value);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * Creates a horizontal line for seperating content on a page
 */
public class HorizontalRule extends HTMLTag
{
    public HorizontalRule() {
        super("hr",false);
    }

    /**
     * @param width a pixel  value for the width of this rule
     * @param noShane indicates whether to have any shade or shadow on this rule
     */
    public HorizontalRule(int width,boolean noShade)
    {
        this();
        addAttribute(new Attribute("width","" + width,false));

        if (noShade)
            addAttribute(new Attribute("noShade"));
    }

    /**
     * @param width a percentage value for the width of this rule
     * @param noShane indicates whether to have any shade or shadow on this rule
     */
    public HorizontalRule(String width,boolean noShade) {
        this();
        addAttribute(new Attribute("width",width,true));

        if (noShade)
            addAttribute(new Attribute("noShade"));
    }
}
```

```java
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

import java.util.Enumeration;

/**
 * The super class for all HTML tag implementions
 */
public class HTMLTag extends Tag
{
    /**
     * A constructor which creates a tag with the given name, and which may either be      ↙
        standalone or a pair.
     * @param name the text to use for this tag (i.e. 'body' would be the value for &lt;body&↙
        gt;)
     * @param closingTag indicates whether this tag has a paired closing tag or is standalone
     *
    */
    public HTMLTag(String name,boolean closingTag) {
        super(name,closingTag);
    }

    /**
     * A constructor which creates a tag set with the given name.
     * @param name the text to use for this tag (i.e. 'body' would be the value for &lt;body&↙
        gt;)
     */
    public HTMLTag(String name) {
        super(name);
    }

    protected String renderOpenTag() {

        StringBuffer output = new StringBuffer();

        //render self open
        output.append("<");
        output.append(getName());

        Enumeration eAttribs = getAttributes().elements();

        while(eAttribs.hasMoreElements())
            output.append(" " + ((Attribute)eAttribs.nextElement()).render());

        output.append(">");

        return output.toString();
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * The basic "page" object, used for all html document rendering
 */
public class HTMLTagDocument extends TagDocument
{
    private Head head;
    private Body body;

    public HTMLTagDocument() {
        super("html","text/html");
    }

    /**
     * @param head set the header section for this page
     */
    public void setHead(Head head) {
        this.head = head;
        resetChildren();
    }

    /**
     * @param body set the main content body section for this page
     */
    public void setBody(Body body) {
        this.body = body;
        resetChildren();
    }

    private void resetChildren()  {
        getRoot().clearChildren();

                try {

        if (head != null)
            getRoot().addChild(head);

        if (body != null)
            getRoot().addChild(body);

              }
              catch(InvalidTagException e) {}
    }

}
```

```java
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;

/**
 * Creates an anchor object with an img tag child
 */
public class HyperlinkedImage extends Anchor
{

    /**
     * @param href the url which to link to
     * @param imgSrc the image url which to load
     */
    public HyperlinkedImage(String href,String imgSrc)
    {
        super("",href,new Image(imgSrc));
    }

    /**
     * @param name a unique id to use for the anchor and image
     * @param href the url which to link to
     * @param imgSrc the image url which to load
     */
    public HyperlinkedImage(String name,String href,String imgSrc)
    {
        this(name,href);

        Image img1 = new Image(imgSrc);
        img1.addAttribute("name",name);
        addChild(img1);
    }

    /**
     * @param name a unique id to use for the anchor and image
     * @param href the url which to link to
     * @param imgSrc the image url which to load
     * @param target the target window or frame which to send the hyperlink action to
     */
    public HyperlinkedImage(String name,String href,String imgSrc,String target)
    {
        this(name,href,imgSrc);

        addAttribute("target",target);
    }

}
```

```
/**
 * @(#)Image.java
 *
 * an element used to insert an image into a page
 */

package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

import java.util.Enumeration;

/**Represents an &lt;img&gt; tag in HTML
*/
public class Image extends HTMLTag
{
    /**
     * Creates the image tag
     *
     * @param src the url to load the image file from
     */
    public Image(String src) {
        super("img",false);
        addAttribute(new Attribute("src",src));
        addAttribute(new Attribute("border","0",false));
    }

    /**
     * @param src the url to load the image file from
     * @param alt the text to display if device does not support image
     */
    public Image(String src,String alt) {
        this(src);
        addAttribute(new Attribute("alt",alt));
    }


    /**
     * @param src the url to load the image file from
     * @param alt the text to display if device does not support image
     * @param width the pixel width of image
     * @param height the pixel height of image
     */
    public Image(String src,String alt,int width,int height) {
        this(src,alt);

        addAttribute(new Attribute("width","" + width,false));
        addAttribute(new Attribute("height","" + height,false));
    }

    /**
     * @param return String returns only the ALT text for this element
     */
    public String getTextOnly() {
            String imgText = getName();

            Enumeration eAttributes = getAttributes().elements();

            Attribute cAttrib;

            while(eAttributes.hasMoreElements()) {

                cAttrib = (Attribute)eAttributes.nextElement();

                if (cAttrib.getName().equalsIgnoreCase("alt")) {
                    imgText = cAttrib.getValue();
                    break;
                }
            }

            return "[" + imgText + "]";

    }

}
```

```
package com.thinairapps.tag.html;

/**
 * A basic form &lt;input&gt; element
 */
public class Input extends FormElement
{
    /**
     * @param type the input type (button, submit, password, etc)
     * @param name the unique id for this input
     */
    public Input(String type,String name)
    {
        super("input",name,false);
        addAttribute("type",type);
    }
    /**
     * @param type the input type (button, submit, password, etc)
     * @param name the unique id for this input
     * @param value the default value
     */
    public Input(String type,String name,String value) {
      this(type,name);
          addAttribute("value",value);
    }

    /**
     * @param value set the default value for this input
     */
        public void setValue(String value) {
            addAttribute("value",value);
        }
}
```

```
package com.thinairapps.tag.html;

/**
 * A form text-type &lt;input&gt; element with a text label
 */
public class LabeledInput extends Input
{
    String label;

    /**
     * @param name the unique id of this input
     * @param label the text label to display
     */
    public LabeledInput(String name,String label) {
        super("text",name);
        this.label = label;
    }

    /**
     * @param name the unique id for this input
     * @param type the input type (button, submit, password, etc)
    * @param value the default value
    * @param label the text label to display
     */
    public LabeledInput(String name,String type,String value,String label) {
        super(type,name,value);
        this.label = label;
    }

    /**
     * @param the text label to use for this input
     */
    public void setLabel(String label) {
        this.label = label;
    }

    public String getLabel() {
        return label;
    }

    public String render() {
        return label + new NonBreakingSpace(1).render() + super.render();
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A tag to use for linking in style-sheets and other content
 */
public class Link extends HTMLTag
{
    /**
    *
    * @param rel indicate the relationship from this document to the target
    * @param type specify the MIME type for the linked document
    * @param href specify the hypertext reference URL of the target document
    */

    public Link (String rel, String type, String href)
    {
        super ("link", false);
        addAttribute ("rel",rel);
        addAttribute ("type",type);
        addAttribute ("href",href);

    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A tag to be used in header sections to indicate some metadata about the content
 */
public class Meta extends HTMLTag
{
    public final static String PROPERTY_NAME = "name";
    public final static String PROPERTY_HTTP_EQUIV = "http-equiv";
    public final static String PROPERTY_USER_AGENT = "user-agent";

    /**
     * @param propertyType the name in the content name/value pair
     * @param propertyValue the value in the content name/value pair
     * @param content the value for the "content" attribute
     */
    public Meta(String propertyType,String propertyValue,String content)
    {
        super("meta",false);
        addAttribute(propertyType,propertyValue);
        addAttribute("content",content);
    }

}
```

```
package com.thinairapps.tag.html;

/**
 * Inserts a blank space into a page
 */
public class NonBreakingSpace extends Text
{
    /**
     * @param number the number of spaces to insert
     */
    public NonBreakingSpace(int number) {
        super("");

        for (int i = 0; i < number; i++)
            setName(getName() + "&nbsp;");
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;
import com.thinairapps.tag.Attribute;

/**
 * An &lt;option&gt; element to be added to a &lt;Select&gt; parent tag
 */
public class Option extends HTMLTag
{
    public Option() {
        super("option");
    }

    /**
    * @param value return the specified value to the form-processing application instead of ↙
        the option contents
    * @param label provide a label for this option
    */
    public Option(String value,String label) {
        this();
        addAttribute("value",value);
         setLabel(label);
    }

    /**
    * @param label set the label
    */

    public void setLabel(String label) {
          try { addChild(new Text(label)); }
          catch(InvalidTagException e) { }
    }

    /**
    * @param child HTMLTag
    */
    public void addChild(HTMLTag child) throws InvalidTagException {
        if (child instanceof Text)
            super.addChild(child);
        else
            throw new InvalidTagException("Option only supports Text or OnEvent children ↙
                tags");
    }

    /**
    * @param selected boolean value for making this item intially selected
    */
    public void setSelected(boolean selected) {
        if (selected)
            addAttribute(new Attribute("selected"));
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;

/**
 * An element used to seperate content section out within a page
 */
public class Paragraph extends HTMLTag
{
    public final static String ALIGN_LEFT = "left";
    public final static String ALIGN_CENTER = "center";
    public final static String ALIGN_RIGHT = "right";
        public final static String ALIGN_JUSTIFY = "justify";

    public Paragraph() {
        super("p");
    }

    public Paragraph(String align) {
        this();
        addAttribute("align",align);
    }

        public void addChild(HTMLTag tag) {
          try { super.addChild(tag); }
          catch(InvalidTagException e) {
            e.printStackTrace();
          }
        }
}
```

```
package com.thinairapps.tag.html;

/**
 * A text form input with an * mask for security
 */
public class PasswordField extends Input
{
    /**
    * @param name specify the name of the parameter to be passed to the form-processing ↙
        applicaiton for this input element
    * @param value specify the initial value for this element
    * @param length specify the maximum number of characters to accept for this element
    */
    public PasswordField (String name, String value, int length) {
        super("password",name,value);
        addAttribute("size","" + length);
    }

    /**
    * @param name specify the name of the parameter to be passed to the form-processing ↙
        applicaiton for this input element
    * @param length specify the maximum number of characters to accept for this element
    */
    public PasswordField (String name, int length) {
        super ("password",name);
        addAttribute("size","" + length);
    }

    /**
    * @param name specify the name of the parameter to be passed to the form-processing ↙
        applicaiton for this input element
    */
    public PasswordField (String name) {
        super ("password",name);
    }
}
```

```java
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * Seperates out text that has been preformatted
 */
public class Pre extends HTMLTag
{
    public Pre() {
        super("pre",true);
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A form element used to reset the form content to its default state
 */
public class ResetButton extends Input
{

    /**
     * @param value specify an alternate label for the reset button
     */
    public ResetButton (String value)
    {
        super("reset","",value);
    }

    /**
     * @param action action taken onClick
     */
    public void setOnClick(String action) {
        addAttribute("onClick",action);
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * Used to set of scripting code from the page content
 */
public class Script extends HTMLTag
{
    /**
     * @param language the scripting language the code is written in
     */
    public Script(String language) {
        super("script",true);

        addAttribute(new Attribute("language",language));
    }

    /**
     * @param code the scripting code to insert into the page
     */
    public void setCode(String code) {
        addChild(new Text(code));
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A form input used to create a list or combo-box
 */
public class Select extends FormElement
{
    /**
     * @param name unique id of this element
     */
    public Select(String name) {
        super("select",name,true);
    }

    /**
     * @param text the displayed text for this entry
     * @param selected indicates if this entry should be selected
     */
    public void addOption(String text,boolean selected)
    {
        Tag option = new Tag("option");
        if (selected)
            option.addAttribute(new Attribute("selected"));

        option.addChild(new Text(text));
        addChild(option);
    }

    /**
     * @param option an Option HTMLTag to add to this select list
     */
        public void addOption(Option option) {
          try { addChild(option); }
          catch(InvalidTagException e) {}
        }


        /**
         * @param value the submitted value for a new option entry
         * @param label the display value for a new option entry
         */
    public void addOption(String value,String label) {
            Option option = new Option(value,label);
            addOption(option);
    }

    /**
     * @param options a set of option entries to add to this select list
     */
    public void setOptions(String[] options) {
        for(int i = 0; i < options.length; i++)
            addOption(new Option(options[i],options[i]));
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;

/**
 * A widget used to create a Body with a form with a select tag and a submit button
 */
public class SelectInputBody extends Body
{

    public SelectInputBody() {
        super();
    }

    /**
     * @param bgColor set the background color of the document
     * @param fgColor set the color of regular text in the document
     * @param linkColor set the color of hypertext links in the document
     */
    public SelectInputBody(String bgColor,String fgColor,String linkColor) {
        super();
        addAttribute("bgColor",bgColor);
        addAttribute("text",fgColor);
        addAttribute("link",linkColor);
        addAttribute("alink",linkColor);
        addAttribute("vlink",linkColor);
    }

    /**
     * @param background specify the URL of an image to be tiled in the document background
     * @param bgColor set the background color of the document
     * @param fgColor set the color of regular text in the document
     * @param linkColor set the color of hypertext links in the document
     */
    public SelectInputBody(String background,String bgColor,String fgColor,String         ↙
        linkColor) {
        super(bgColor,fgColor,linkColor);
        addAttribute("background",background);
    }

    /**
     * @param href the url to submit the form to
     * @param label the label to use for the select element
     * @param name the unique id of the select element
     * @param optionValues the array of name/value pairs to use for the option elements
     * @param align the Paragraph alignment value to use for the content
     */
    public void buildPage(String href,String label,String name,String[][] optionVals,String   ↙
        align) throws InvalidTagException {

        Option[] options = new Option[optionVals.length];

        for (int i = 0; i < optionVals.length; i++)
            options[i] = new Option(optionVals[i][1],optionVals[i][0]);

        buildPage(href,label,name,options,align);
    }

    /**
     * @param href the url to submit the form to
     * @param label the label to use for the select element
     * @param name the unique id of the select element
     * @param options the array of Option tags to add to the select list
     * @param align the Paragraph alignment value to use for the content
     */
    public void buildPage(String href,String label,String name,Option[] options,String        ↙
        align) throws InvalidTagException {

        Paragraph p = new Paragraph(align);
        p.addChild(new Text(label));
        p.addChild(new Break());

                Form form = new Form("form1",href,"GET");

        Select select = new Select(name);
        Option cOpt = null;

        for (int i = 0; i < options.length; i++)
            select.addOption(options[i]);


                form.addChild(select);
                form.addChild(new SubmitButton("Submit"));

        p.addChild(form);

        addChild(p);
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;

/**
 * A widget used to create a Body with a form with a text input and a submit button
 */
public class SingleInputBody extends Body
{

    public SingleInputBody ()
    {
        super();
    }

    /**
     * @param href url to submit form to
     * @param label text label for input tag
     * @param name unique id for this form input
     * @param buttonLabel label for submit button
     */
    public void buildPage (String href, String label, String name, String buttonLabel)
    {

        Paragraph p = new Paragraph();
        Form form = new Form("form1",href,"GET");
        LabeledInput input = new LabeledInput(name,"text","",label);
        form.addFormElement(input);
        form.addFormElement(new SubmitButton(buttonLabel));
        p.addChild(form);
        addParagraph(p);

    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.InvalidTagException;

/**
 * A widget used to create a Body tag with a form with a text input and a submit button
 */
public class SingleInputPage extends Body
{

    public SingleInputPage() {
        super();
    }


    /**
     * @param href url to submit form to
     * @param label text label for input tag
     * @param name unique id for this form input
     * @param buttonLabel label for submit button
     */
    public void buildPage(String href,String label,String name,String buttonLabel) {

        Paragraph p = new Paragraph();

                Form form = new Form("form1",href,"GET");
        LabeledInput input = new LabeledInput(name,"text","",label);
                form.addFormElement(input);
                form.addFormElement(new SubmitButton(buttonLabel));
                p.addChild(form);
        addParagraph(p);
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A form element used to trigger the submit action
 */
public class SubmitButton extends Input
{
    /**
     * @param value the display text on the button
     */
    public SubmitButton(String value) {
        super("submit","submit",value);
    }
    public SubmitButton(String value, String name) {
        super("submit", name, value);
    }

    /**
     * @param action the scripting action to trigger when the button is clicked
     */
    public void setOnClick(String action) {
        addAttribute("onClick",action);
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A &lt;table@gt; HTML element to display tabular structured content
 */
public class Table extends HTMLTag
{
    /**
     * @param border an int value indicated the thickness of the table border
     */
    public Table (int border)
    {
        super("table",true);
        addAttribute(new Attribute("border","" + border,false));
    }

    /**
     * @param border an int value indicated the thickness of the table border
     * @param cellpadding an int value indicated space padding within each table cell
     * @param cellspacing an int value indicated spacing between each table cell
     */
    public Table(int border,int cellpadding,int cellspacing) {
        this(border);
        addAttribute(new Attribute("cellpadding","" + cellpadding,false));
        addAttribute(new Attribute("cellspacing","" + cellspacing,false));
    }

    /**
     * @param border an int value indicated the thickness of the table border
     * @param cellpadding an int value indicated space padding within each table cell
     * @param cellspacing an int value indicated spacing between each table cell
     * @param width sets the width of the table in pixels
     * @param height sets the height of the table in pixels
     */
    public Table(int border,int cellpadding,int cellspacing,int width,int height) {
        this(border,cellpadding,cellspacing);
        addAttribute(new Attribute("width","" + width,false));
        addAttribute(new Attribute("height","" + height,false));
    }
    /**
     * @param border an int value indicated the thickness of the table border
     * @param cellpadding an int value indicated space padding within each table cell
     * @param cellspacing an int value indicated spacing between each table cell
     * @param width sets the width of the table in percentage
     * @param height sets the height of the table in percentage
     */
    public Table(int border,int cellpadding,int cellspacing,String width,String height) {
        this(border,cellpadding,cellspacing);
        addAttribute(new Attribute("width",width));
        addAttribute(new Attribute("height",height));
    }

}
```

```
package com.thinairapps.tag.html;

/**
 * &lt;td&gt; - an individual cell for a table
 */
public class TableCell extends TableElement
{
    public TableCell() {
        super("td");
    }

    /**
     * @param width pixel width of cell
     * @param height pixel height of cell
     *
     */
    public TableCell(int width,int height) {
        super("td",width,height);
    }

    /**
     * @param width pixel width of cell
     * @param height pixel height of cell
     * @param bgColor hex or named color for background of cell
     */
    public TableCell(int width,int height,String bgColor) {
        super("td",width,height,bgColor);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A super class for all table cells, headers rows, etc
 */
public class TableElement extends HTMLTag
{
    /**
     * @param tag name of Table
     */
    public TableElement(String tag) {
        super(tag,true);
    }

    /**
     * @param tag name of Table
     * @param width width of Table
     * @param height height of Table
     */
    public TableElement(String tag,int width,int height) {
        this(tag);
        addAttribute(new Attribute("width","" + width,false));
        addAttribute(new Attribute("height","" + height,false));
    }

    /**
     * @param tag name of Table
     * @param width width of Table
     * @param height height of Table
     * @param bgColor define the background color for the entire Table
     */
    public TableElement(String tag,int width,int height,String bgColor) {
        this(tag,width,height);
        addAttribute("bgColor",bgColor);
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A table cell to use in the header row
 */
public class TableHeader extends TableElement
{
    public TableHeader() {
        super("th");
    }

    /**
     * @param width set the width of the cell to X pixels or a percentage of the table width
     * @param height define the height, in pixels, for this cell
     */
    public TableHeader (int width,int height) {
        super("th",width,height);
    }


    /**
     * @param width set the width of the cell to X pixels or a percentage of the table width
     * @param height define the height, in pixels, for this cell
     * @param bgColor define the background color for the cell
     */
    public TableHeader ( int width, int height, String bgColor) {
        super("th",width,height,bgColor);
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * &lt;tr&gt;- the element to use for each row of the table
 */
public class TableRow extends TableElement
{
    public TableRow () {
        super("tr");
    }

    public TableRow (int width, int height) {
        super("tr",width,height);
    }

    public TableRow (int width, int height, String bgColor) {
        super("tr",width,height,bgColor);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A node used to wrap any text content for a page
 */
public class Text extends HTMLTag
{

    public Text(String text) {
        super(text,false);
    }

    public String getTextOnly() {
        return getName();
    }

    public String render() {
        return getName();
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A multiline, scrollable text area input form element
 */
public class TextArea extends FormElement
{
    /**
     * @param name the unique id of this element
     * @param rows the number of rows
     * @param cols the number of columns
     */
    public TextArea(String name,int rows,int cols) {
        super("textarea",name,true);

        addAttribute("rows","" + rows);
        addAttribute("cols","" + cols);
    }

    /**
     * @param text set the text value for this element
     */
    public void setValue(String text) {
        getChildren().removeAllElements();
        getChildren().addElement(new Text(text));
    }
}
```

```
package com.thinairapps.tag.html;

/**
 * A single text input form element
 */
public class TextField extends Input
{
    /**
     * @param name specify the name of the parameter that is passed to the form-processing ↙
        application for this input element
     * @param value specify the intial value for this element
     * @param length specify the maximum number of characters to accept for this element
     */

    public TextField (String name, String value, int length) {
        super("text",name,value);
        addAttribute("size","" + length);
    }

    /**
     * @param name specify the name of the parameter that is passed to the form-processing ↙
        application for this input element
     * @param length specify the maximum number of characters to accept for this element
     */
    public TextField (String name, int length) {
        super ("text",name);
        addAttribute("size","" + length);
    }

    /**
     * @param name specify the name of the parameter that is passed to the form-processing ↙
        application for this input element
     */
    public TextField (String name) {
        super ("text",name);
    }
}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A utility wrapper class for applying styles to text content
 */
public class TextStyle extends HTMLTag
{
    public final static int PLAIN = 0;
    public final static int BOLD = 1;
    public final static int ITALIC = 2;
    public final static int UNDERLINE = 3;
    public final static int TT = 4;

    public final static int H1 = 5;
    public final static int H2 = 6;
    public final static int H3 = 7;
    public final static int H4 = 8;
    public final static int H5 = 9;
    public final static int H6 = 10;

    int style = PLAIN;

    public TextStyle(int style) {
        super("",true);
        this.style = style;
    }

    /**
     * @param style the style CONSTANT to apply
     * @param child the htmltag to wrap the style around
     */
    public TextStyle(int style,HTMLTag child) {
        super("",true);
        this.style = style;
        addChild(child);
    }

    protected String renderOpenTag() {
        if (style == BOLD)
            return "<b>";
        else if (style == ITALIC)
            return "<i>";
        else if (style == UNDERLINE)
            return "<u>";
        else if (style == TT)
            return "<tt>";
        else if (style == H1)
            return "<h1>";
        else if (style == H2)
            return "<h2>";
        else if (style == H3)
            return "<h3>";
        else if (style == H4)
            return "<h4>";
        else if (style == H5)
            return "<h5>";
        else if (style == H6)
            return "<h6>";
        else
            return "";
    }

    protected String renderCloseTag() {
        if (style == BOLD)
            return "</b>";
        else if (style == ITALIC)
            return "</i>";
        else if (style == UNDERLINE)
            return "</u>";
        else if (style == TT)
            return "</tt>";
        else if (style == H1)
            return "</h1>";
        else if (style == H2)
            return "</h2>";
        else if (style == H3)
            return "</h3>";
        else if (style == H4)
            return "</h4>";
        else if (style == H5)
            return "</h5>";
        else if (style == H6)
            return "</h6>";
        else
            return "";
    }

}
```

```
package com.thinairapps.tag.html;

import com.thinairapps.tag.*;

/**
 * A &lt;title&gt; tag to use in the header area of a page
 */
public class Title extends HTMLTag
{
    /**
     * @param title specify the title of the HTML doc
     */
    public Title(String title)  throws InvalidTagException {
        super("title",true);
        addChild(new Text(title));
    }
}
```

```
package com.thinairapps.tag.html.pqa;

import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;

/**
 * A Palm anchor with the BUTTON attribute, indicated it should
 * be rendered as a button
 *
 * @param text the Text for the button
 * @param url the URL that is associated with the button
 */

public class AnchorButton extends Anchor
{
    public AnchorButton (String text, String url)
    {
        super ("",url,new Text(text));

        addAttribute (new Attribute("BUTTON"));
    }
}
```

```
package com.thinairapps.tag.html.pqa;

import com.thinairapps.tag.html.*;

/**
 * A Palm form element that shows a calendar view
 *
 */
public class DatePicker extends Input
{
    /**
    * @param name Name for DatePicker
    */
    public DatePicker (String name)
    {
        super ("datepicker", name);
    }
}
```

```
package com.thinairapps.tag.html.pqa;

import com.thinairapps.tag.html.Meta;

/**
 * a palm header entry for controlling the displayed text in the HISTORY cache
 *
 */
public class HistoryListMeta extends Meta
{
    /**
     * @param text text for HistoryListMeta tag
     */
    public HistoryListMeta (String text)
    {
        super("name","historylisttext",text);
    }
}
```

```
package com.thinairapps.tag.html.pqa;

/**
 * Some constants to use in url building.  See Palm's PQA documentation for more information.
 */
public class PalmConstants
{
    /**This get the unique device ID for the device making the request.
    */
    public final static String DEVICE_ID = "%deviceid";
    /**This get the ZIP code for the nearest radio tower.
    */
    public final static String ZIP_CODE = "%zipcode";

}
```

```java
package com.thinairapps.tag.html.pqa;

import com.thinairapps.tag.html.Meta;

/**
 * The meta tag to insert into a header to indicate the page should not
 * be "scraped" or "clipped" by the proxy, since it is already palm friendly
 */
public class PalmContentMeta extends Meta
{
    public PalmContentMeta ()
    {
        super("name","PalmComputingPlatform","true");
    }
}
```

```
package com.thinairapps.tag.html.pqa;

import com.thinairapps.tag.html.*;

/**
 * A form element that display a time widget in a Palm PQA
 */
public class TimePicker extends Input
{
    /**
     * @param name specify the name of the element
     */
    public TimePicker (String name)
    {
        super ("timepicker", name);
    }
}
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * Represents an ACTION tag.
 * binds a label, an optional image, and a ActionTask to the user's agent's navigational user↙
   interface
 * when the action is selected the indicated ActionTask takes place
 */
public class Action extends HDMLTag {

    /** defines the acceptable action types */
    public static class Type {
        private String name;
        private Type(String n) { name = n; }
    } // end Type


    public static final Type SOFT1  = new Type("SOFT1");
    public static final Type SOFT2  = new Type("SOFT2");
    public static final Type SOFT3  = new Type("SOFT3");
    public static final Type SOFT4  = new Type("SOFT4");
    public static final Type SOFT5  = new Type("SOFT5");
    public static final Type SOFT6  = new Type("SOFT6");
    public static final Type SOFT7  = new Type("SOFT7");
    public static final Type SOFT8  = new Type("SOFT8");
    public static final Type ACCEPT = new Type("ACCEPT");
    public static final Type PREV   = new Type("PREV");
    public static final Type HELP   = new Type("HELP");


    private Type type;
    private ActionTask task;

    /**
     * create a new HDML action
     * @param type one of Type instances above
     * @param task specifies the way in which the action is carried out (i.e. invoke ↙
       subprocedure)
     */
    public Action(Type type, ActionTask task) {
        this(type);

        addAttribute( new Attribute("TASK", task.render(), false) );
        this.task = task;
    }

    /**
     * create an action with no Task useful for Choice lists
     * @param type one of Type instances above
     */
    public Action(Type type) {
        super("ACTION", false);
        addAttribute( new Attribute("TYPE", type.name, false) );
        this.type = type;
    }


    /**
     * create a new HDML action with a destination, a label
     * and no ActionTask
     * @param type one of Type instances above
     * @param label String name to map to button invoking this action
     * @param dest String URL destination to go to when Action is executed
     */
    public Action(Type type, String label, String dest) {
        this(type);
        setLabel(label);

        ActionTask task = new ActionTask(ActionTask.GOSUB);
        task.setDest(dest);
        addAttribute( new Attribute("TASK", task.render(), false) );
        this.task = task;
    }


    /**
     * create a new HDML action with a label
     * @param type one of Type instances above
     * @param task specifies the way in which the action is carried out (i.e. invoke ↙
        subprocedure)
     * @param dest String URL destination to go to when Action is executed
     */
    public Action(Type type, ActionTask task, String label) {
        this(type, task);
        setLabel(label);
    }


    /**
     * create a new HDML action with a label and an image
     * @param type one of Type instances above
     * @param task specifies the way in which the action is carried out (i.e. invoke ↙
        subprocedure)
     * @param label String name to map to button invoking this action
     * @param image image tag to render Action (if supported by phone)
     */
    public Action(Type type, ActionTask task, String label, ImageTag image) {
        this(type, task);
        setLabel(label);
        setImage(image);
    }


    /**
     * set the label option
     * the text to display for this action - try to keep to <= 6 characters
     * @param label String to map to button executing Action
     */
    public void setLabel(String label) {
        addAttribute( new Attribute("LABEL", label) );
    }


    /**
     * set the url of the image to display for this action
     * @param image image tag to render Action (if supported by phone)
     */
    public void setImage(ImageTag image) {
        addAttribute( new Attribute("IMAGE", image.getSrc()) );
    }

    /**
     * @return Type the type for this action
     */
    public Type getType() { return type; }

    /**
     * @return ActionTask the ActionTask for this action
     */
    public ActionTask getActionTask() { return task; }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;
import java.util.*;

/**
 * Represents a ActionTask bound to a ChoiceEntry within a Choice card.
 * <p>
 * Different ActionTask types have different options -- see the HDML spec
 * for a complete account of which options are relevant for which ActionTask types
 */
public class ActionTask extends HDMLTag {

    /** This inner class defines the acceptable types for Action Tasks */
    public static class Type {
        private String name;
        private Type(String n) { name = n; }
        public String toString(){return name;}
    } // end Type

    public static final Type GO       = new Type("GO");
    public static final Type GOSUB    = new Type("GOSUB");
    public static final Type PREV     = new Type("PREV");
    public static final Type RETURN   = new Type("RETURN");
    public static final Type CANCEL   = new Type("CANCEL");
    public static final Type POST     = new Type("POST");
    public static final Type CALL     = new Type("CALL");
    public static final Type NOOP     = new Type("NOOP");

    private static final String mismatch = "this option is not valid for this ActionTask ↙
        Type";
    private Type type;


    /**
     * create a ActionTask of a specific type
     * @param t create an ActionTask of this type
     */
    public ActionTask(Type t) {
        super(t.name, false);
        type = t;
    }



    /**
     * set the dest option
     * the URL of the card to display or invoke in the GO, GOSUB, RETURN, or CANCEL ↙
        ActionTasks
     * @param url String destination for this action task
     */
    public void setDest(String url) {

        if (type == GO || type == GOSUB || type == RETURN || type == CANCEL)
            ; // ok
        else
            throw new InvalidHDMLException(mismatch);

        addAttribute( new Attribute("DEST", url, false) );
    }


    /**
     * specifies a single name=value variable pair to set in the current (in the case of GO)
     * or sub (in the case of SUB) activity.
     * each pair will be appended to the URL-style vars String value
     * @param name String variable name
     * @param value String variable default value
     */
    public void setVar(String name, String value) {

        if (type == GO || type == GOSUB)
            ; // ok
        else
            throw new InvalidHDMLException(mismatch);

        Attribute vars = getAttribute("VARS");
        String varString = name + "=" + value + "&";

        if (vars != null)
            varString = vars.getValue() + varString;

        // overwrite the old VARS Attribute
        addAttribute(new Attribute("VARS", varString));
        vars = null;
    }



    /**
     * set the RECEIVE option
     * when invoking a card with the GOSUB ActionTask the RECEIVE option specifies the names ↙
       of the
     * variables to assign the return values to
     * based on position
     * @param variables list of variables to receive values from a GOSUB ActionTask
     */
    public void setReceiveList(String variables[]) {

        if (type != GOSUB)
            throw new InvalidHDMLException(mismatch);

        StringBuffer sb = new StringBuffer(128);
        for (int i = 0; i < variables.length; i++) {
            sb.append( variables[i] );
            sb.append(";");
        }
        String s = sb.toString().trim();
        // remove that trailing ;
        s = s.substring(0, s.length() - 1);

        addAttribute( new Attribute("RECEIVE", s) );
    }



    /**
     * set the RETVALS option
     * when returning from a sub-activity with the RETURN ActionTask, the RETVALS option       ↙
       specifies
     * the values to return to the invoking activity
     * values are positional
     * @param retVals array of values to return from the task
     */
    public void setRetvals(String retVals[]) {

        if (type != RETURN)
            throw new InvalidHDMLException(mismatch);

        StringBuffer sb = new StringBuffer(128);
        for (int i = 0; i < retVals.length; i++) {
            sb.append( retVals[i] );
            sb.append(";");
        }
        String s = sb.toString().trim();
        // remove that trailing ;
        s = s.substring(0, s.length() - 1);
        addAttribute( new Attribute("RETVALS", s) );
    }



    /**
     * set the NEXT option
     * the next option specifies the destination to go to after the sub-activity returns
     * @param destination URL to go to when the action is executed
     */
  public void setNext(String destination) {

    if (type != GOSUB)
      throw new InvalidHDMLException(mismatch);

    addAttribute( new Attribute("NEXT", destination) );
  }


    /**
     * set the CANCEL option
     * when invoking a sub-activity with the GOSUB ActionTask, the CANCEL option specifies
        the
     * destination to go to if the sub-activity is cancelled
     * @param destination URL to go to when the action is executed
     */
  public void setCancel(String destination) {

    if (type != GOSUB)
      throw new InvalidHDMLException(mismatch);

    addAttribute( new Attribute("CANCEL", destination) );
  }


    /**
     * the SENDREFERER option specifies whether the user agent should indicate the URL of the
     * referring deck when requesting the DEST, NEXT, CANCEL decks from the server
     * @param boolean if true then send referer URL
     */
  public void setSendReferer(boolean b) {

    if (type == GO || type == GOSUB)
      ; // ok
    else
      throw new InvalidHDMLException(mismatch);

    String val = ((b) ? "true" : "false");
    addAttribute( new Attribute("SENDREFERER", val, false) );
  }


    /**
     * set the friend option to indicate that the sub-activity is friendly
     * @param boolean if true then the sub-activity is friendly
     */
  public void setFriend(boolean b) {

    if (type != GOSUB)
      throw new InvalidHDMLException(mismatch);

    String val = ((b) ? "true" : "false");
    addAttribute( new Attribute("FRIEND", val, false) );
  }



    /**
     * set the clear option to indicate that the sub-activity is clearly
     * used by RETURN and CANCEL to unset all calling activity's variables
     * @param boolean if true clear the subactivity
     */
  public void setClear(boolean b) {

    if (type == RETURN || type == CANCEL)
      ; // ok;
    else
      throw new InvalidHDMLException(mismatch);

    String val = ((b) ? "true" : "false");
    addAttribute( new Attribute("CLEAR", val, false) );
  }


    /**
     * set the NUMBER option - specifies the phone number for a CALL ActionTask
     * @param value the phone number String
     */
  public void setNumber(String value) {

    if (type != CALL)
      throw new InvalidHDMLException(mismatch);

    addAttribute( new Attribute("NUMBER", value) );
  }


    /**
     * @return String typename
     */
  public String getTypename() { return type.name; }


    /**
     * override here so you can embed this tag within the attribute list of an Action
     */
    protected String renderOpenTag() {
      StringBuffer output = new StringBuffer();
      output.append(name);

      Enumeration enum = attributes.elements();
      while( enum.hasMoreElements() )
    output.append(" " + ((Attribute) enum.nextElement()).render());

      return output.toString();
    }

 } // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * The &lt;anchor&gt; element anchors a task to a string of formatted text, often called a ↙
   link.
 * You can specify a link within any formatted text or image.  Any one of ActionTasks must be
 * bound to the Anchor for it to be bound to a button on the device and perform some action ↙
   when
 * selected.
 */
public class Anchor extends HDMLTag {

    /**
     * Create an Anchor without a TAKS or DEST
     */
    public Anchor() { super("A",true); }

    /**
     * Create an Anchor of the given type, destination, and text label
     * @param type any one of 8 ActionTasks bound to the link
     * @param dest destination for the action type
     * @param text label for link
     */
    public Anchor(ActionTask.Type type,String dest, Text text) {
        super("A",true);
        addAttribute("TASK",type.toString());
        addAttribute("DEST",dest);
        addChild(text);
    }

    /**
     * Set the task for this anchor
     * @param type specifies the way in which the action is carries out (i.e. invoke ↙
        subprocedure)
     */
    public void setTask(ActionTask.Type type) {
        addAttribute("TASK",type.toString());
    }

    /**
     * Set the destination of this action
     * @param dest URL destination
     */
    public void setDest(String dest) {
        addAttribute("DEST",dest);
    }

} // end
```

```
package com.thinairapps.tag.hdml;

/**
 * A line-break tag akin to &lt;BR&gt; in HTML
 */
public class Break extends HDMLTag {
    /** create a new line-break tag */
    public Break() {
        super("BR",false);
    }
} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * The base class for all HDML cards
 */
public abstract class Card extends HDMLTag {

    /**
     * Build a new Card
     * @param typeName the type of card to build
     * @param boolean should the card append a closing tag i.e. <something /> or leave the
     * tag open <something> and wait for a </something> later
     */
  protected Card(String typeName, boolean closingTag) {
    super(typeName, closingTag);
  }


    /**
     * set the name option of this card
     * if the card has a name then it can be referreed to as a fragment in a destination
     * @param name the String name of the card
     */
  public void setName(String name) {
    addAttribute( new Attribute("NAME", name) );
  }


    /**
     * set the title option of the card
     * if no title is specified the first text line of the card is used as the title
     * title is used for: a suggested bookmark name
     *                    a text entry prompt
     * @param title String title to be displayed by some browsers at the top of the card
     */
  public void setTitle(String title) {
    addAttribute( new Attribute("TITLE", title) );
  }


    /**
     * @return String title of the card
     */
  public String getTitle() {
    Attribute att = getAttribute("TITLE");
    return att.getValue();
  }

    /**
     * specify a URL to use when bookmarking the card
     * the default is the URL of the current card
     * this option is used to force the bookmark to go to another card (like a NODISPLAY
       card) that sets up valirables
     * instead of the current card
     * @param url String to use when bookmarking the card
     */
  public void setBookmark(String url) {
    addAttribute( new Attribute("BOOKMARK", url) );
  }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * Represents the CHOICE card.
 * <p>
 * Lets users pick from a list of choices - the initial display content is shown to the user
 * followed by a list of choices.  Each choice can have one line of formatted text
 * text defaults to line mode but may optionally be wrapped
 */
public class ChoiceCard extends Card {

    public static class Method {
        private String name;
        private Method(String n) { name = n; }
    }
    public static final Method NUMBER = new Method("number");
    public static final Method ALPHA  = new Method("alpha");


    /**
     * create a choice card with a line of text
     * @param text String to display above the list of choices
     */
    public ChoiceCard(String text) {
        super("CHOICE", true);
        try {
            addChild( new FormattedLine(text, true) );
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }


    /**
     * create a choice card with no text and all defaults
     */
    public ChoiceCard() {
        super("CHOICE", true);
    }



    /**
     * add the display text to this ChoiceCard
     * @param text the label
     */
    public void addText(FormattedLine text) {
        try {
            addChild(text);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }



    /**
     * set the method option for this ChoiceCard
     * @param method defaults to method=NUMBER
     */
    public void setMethod(Method method) {
        addAttribute( new Attribute("METHOD", method.name) );
    }


    /**
     * set the KEY option
     * @param key indicates the name of the variable in the current activity to be set by ↙
        this entry
     */
    public void setKey(String key) {
        addAttribute( new Attribute("KEY", key) );
    }


    /**
     * set the DEFAULT
     * @param def indicates the name of the variable in the current activity to be set by ↙
        this entry
     */
    public void setDefault(String def) {
        addAttribute( new Attribute("DEFAULT", def) );
    }


    /**
     * set the key and default options
     * indicates the name of the variable to be set to the choice entry value
     * when that ce is picked
     * an entry is picked when any action (ACCEPT, PREV, SOFT*) is selected
     *
     * default indicates the default value of the variable key - when the card is entered
     * if the variable keu is not set it will be assigned the default value
     * otherwise default is ignored
     * @param key indicates the name of the variable in the current activity to be set by ↙
        this entry
     * @param def indicates the name of the variable in the current activity to be set by ↙
        this entry
     */
  public void setKey(String key, String def) {
    addAttribute( new Attribute("KEY", key) );
    addAttribute( new Attribute("DEFAULT", def) );
  }


    /**
     * ikey indicates the name of the variable to be set to the entru index when
     * an entry is picked - the entry index is the positiion of the currently-selected
     * choice entry in the choice card
     * an index of 0 indicates that no choice entry is selected
     *
     * idefault indicates the default selected entry
     * if ikey is not specified, idefault will be applied every time the card is entered
     * @param key indicates the name of the variable in the current activity to be set by ↙
        this entry
     * @param def indicates the name of the variable in the current activity to be set by ↙
        this entry
     */
  public void setIKey(String key, String def) {
    addAttribute( new Attribute("IKEY", key) );
    addAttribute( new Attribute("IDEFAULT", def) );
  }

    /**
     * add a choice entry to this card
     * @param ce the ChoiceEntry to add
     */
  public void addChoiceEntry(ChoiceEntry ce) {
    try {
      addChild(ce);
    } catch (Exception e) {
      throw new InvalidHDMLException(e.getMessage());
    }
  }
```

```
    /**
     * add an action to this Card
     * when an action is specified for a card it overrides any deck actions of that type
     * while the card is visible
     *
     * N.B. all actions must precede any line of text in the hdml document
     * @param action to bind to some button
     */
    public void addAction(Action action) {
        try {
            addChild(action);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * choice cards must have one or more choices
     * implement that checking here
     */
    public String render() {
        if (getChildren().size() == 0)
            throw new InvalidHDMLException("Choice cards must have >= 1 Choice Entry");

        return super.render();
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;
import java.util.*;

/**
 * Represents a single CE choice entry within a Choice card.
 * In addition to text you can specify a value to be assigned to the variable
 * named in the parent choice card's key option
 */
public class ChoiceEntry extends HDMLTag {

    /**
     * create a choice Entry with all defaults
     * @param text label for this Choice Entry
     */
    public ChoiceEntry(String text) {
        this();
        try {
            addChild( new FormattedLine(text) );
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * create a choice Entry with a name and dest
     * @param text label for this Choice Entry
     * @param dest URL destination to go to when the choice is selected
     */
    public ChoiceEntry(String text, String dest) {
        this();
        try {
            addChild( new FormattedLine(text) );
            setDest(dest);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * create a choice Entry with a name, dest, and value
     * @param text label for this Choice Entry
     * @param dest URL destination to go to when the choice is selected
     * @param value to be assigned to the variable named in the choice card's key option
     */
    public ChoiceEntry(String text, String dest, String value) {
        this();
        try {
            addChild( new FormattedLine(text) );
            setDest(dest);
            setValue(value);

        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * create a choice Entry with no text, value, or destination
     */
    public ChoiceEntry() {
        super("CE", false);
    }
    /**
     * add the display text to this ChoiceEntry
     * @param text String label for this choice
     */
    public void addText(FormattedLine text) {
        try {
            addChild( text );
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * set the value to be assigned to the variable named in the choice card's key option
     * @param value
     */
    public void setValue(String value) {
        addAttribute( new Attribute("VALUE", value) );
    }


    /**
     * adds an ActionTask to this ChoiceEntry and sets the dest attribute
     * @param dest URL destination to go to when the choice is selected
     */
    public void setDest(String dest) {
        if (getAttribute("TASK") == null) addAttribute( new Attribute("TASK", "GOSUB") );
        addAttribute( new Attribute("DEST", dest) );
    }


    /**
     * set the ActionTask associated with this ChoiceEntry
     * be sure to include all attributes of the Task as attributes of the ChoiceEntry
     * @param task ActionTask
     */
    public void setActionTask(ActionTask task) {
        addAttribute( new Attribute("TASK", task.getTypename(), false) );
        Enumeration enum = task.getAttributes().elements();

        while (enum.hasMoreElements())
            addAttribute((Attribute) enum.nextElement());
    }

} // end
```

```java
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * the DISPLAY card is used to give information for the user to read
 * DISPLAY cards can also contain actions
 */
public class DisplayCard extends Card {

    /**
     * create a display card
     */
    public DisplayCard() {
        super("DISPLAY", true);
    }


    /**
     * create a display card
     * @param text String label
     */
    public DisplayCard(String text) {
        this();
        addText( new FormattedLine(text) );
    }



    /**
     * add an action to this Card
     * when an action is specified for a card it overrides any deck actions of that type
     * while the card is visible
     *
     * N.B. all actions must precede any formatted lines in the hdml document
     * @param action to bind to some button on the device
     */
    public void addAction(Action action) {
        try {
            addChild(action);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * add a line of formatted text to this DISPLAY card
     * @param line text to add to card
     */
    public void addText(FormattedLine line) {
        try {
            addChild(line);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * Represents the ENTRY card.
 * <p>
 * Lets users input an optionally formatted character string - the display content
 * is shown to the user, followed by an area to input characters
 */
public class EntryCard extends Card {


    /**
     * create an entry card with String text
     * @param text String text to add to top of card
     */
    public EntryCard(String text) {
        super("ENTRY", true);
        addText( new FormattedLine(text) );
    }

    /**
     * create an entry card with String text
     * @param text String text to add to top of card
     * @param key indicates the name of the variable in the current activity to be set by ↙
        this entry
     * @param action to bind to some button on the device
     */
    public EntryCard(String text, String key, Action action) {
        super("ENTRY", true);
        setKey(key);
        addAction(action);
        addText( new FormattedLine(text) );
    }


    /**
     * create a choice card with no text, key, or actioni
     */
    public EntryCard() {
        super("ENTRY", true);
    }



    /**
     * set the KEY option
     * @param key value of the KEY attribute
     */
    public void setKey(String key) {
        addAttribute( new Attribute("KEY", key) );
    }


    /**
     * set the DEFAULT option
     * @param def indicates the name of the variable in the current activity to be set by ↙
        this entry
     */
    public void setDefault(String def) {
        addAttribute( new Attribute("DEFAULT", def) );
    }



    /**
     * set the KEY and DEFAULT options
     *
     * indicates the name of the variable in the current activity to be set by this entry
     *
     * default indicates the default value of the variable key - when the card is entered
     * if the variable keu is not set it will be assigned the default value
     * otherwise default is ignored
     *
     * N.B. default must conform to the format optioin if that is set
     * @param key value of the KEY attribute
     * @param def indicates the name of the variable in the current activity to be set by
       this entry
     */
    public void setKey(String key, String def) {
        addAttribute( new Attribute("KEY", key) );
        addAttribute( new Attribute("DEFAULT", def) );
    }


    /**
     * set the FORMAT optiion - used to specify a format for user input entries
     * @see http://www.w3c.org/TR/hdml20-6.html for format codes
     * @param format the format String, i.e. 'm*' for all alphanumeric defaulting to
       lower-case
     */
    public void setFormat(String format) {
        addAttribute( new Attribute("FORMAT", format) );
    }



    /**
     * set the NOECHO option on the text field
     * @param b if true then chars will not be echoed
     */
    public void setNoEcho(boolean b) {
        String val = ((b) ? "true" : "false");
        addAttribute( new Attribute("NOECHO", val, false) );
    }



    /**
     * set the EMPTYOK option on the text field
     * @param b if true then an empty input will be accepted
     */
    public void setEmptyOK(boolean b) {
        String val = ((b) ? "true" : "false");
        addAttribute( new Attribute("EMPTYOK", val, false) );
    }

    /**
     * add an action to this Card
     * when an action is specified for a card it overrides any deck actions of that type
     * while the card is visible
     *
     * N.B. all actions must precede any line of text in the hdml document
     * @param action to bind to some button on the device
     */
    public void addAction(Action action) {
        try {
            addChild(action);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }


    /**
     * add a line of formatted text to this DISPLAY card
     * @param text String text to add to top of card
     */
    public void addText(FormattedLine line) {
        try {
            addChild(line);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

} // end
```

```java
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * a subclass of DISPLAY card for displaying errors to the user
 */
public class ErrorCard extends DisplayCard {

    /**
     * create an error card
     *
     * @param String error message
     */
  public ErrorCard(String error) {
    super();
    addText( new FormattedLine(error, true) );
  }

    /**
     * create an error card
     *
     * @param String error message
     * @param String path for ok button
     * @param String label on ok button
     */
  public ErrorCard(String error, String path, String label) {
    super();
    ActionTask task = new ActionTask(ActionTask.GO);
    task.setDest(path);
    addAction( new Action(Action.ACCEPT, task, label) );

    addText( new FormattedLine(error, true) );
  }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * A line of text, with formatting directives, to add to the card
 */
public class FormattedLine extends HDMLTag {

    /** Instances of this class represent the acceptable alignments */
    public static class Alignment { private Alignment() { ; } }
    public static Alignment LEFT = new Alignment();
    public static Alignment CENTER = new Alignment();
    public static Alignment RIGHT = new Alignment();

    private String lineFormat;
    private String alignmentFormat;

    /**
     * create a line with no text - a break
     */
    public FormattedLine() {
        super("<BR>", false);
    }


    /**
     * create a line of formatted text
     * @param content String content of the text line
     * @param align one of three alignment instances above
     */
    public FormattedLine(String content, Alignment align) {
        super(content, false);

        if (align == RIGHT)
            alignmentFormat = "<RIGHT>";
        else if (align == CENTER)
            alignmentFormat = "<CENTER>";
    }


    /**
     * create a line of formatted text with default alignment (LEFT)
     * @param content String content of the text line
     * @param wrap if true then the text will wrap around the screen
     */
    public FormattedLine(String content, boolean wrap) {
        super(content, false);
        lineFormat = (wrap) ? "<WRAP>" : "<LINE>";
    }


    /**
     * create a line of formatted text with all defaults
     * @param content String content of the text line
     */
    public FormattedLine(String content) {
        super(content, false);
        // lineFormat = alignmentFormat = null;
    }


    /**
     * set the line format mode
     * @param b if true then the text will wrap around the screen
     */
    public void setWrap(boolean b) {
        lineFormat = (b) ? "<WRAP>" : "<LINE>";
    }
    /**
     * set the alignment format mode
     * @param align one of three alignment instances above
     */
    public void setAlignment(Alignment align) {
        if (align == RIGHT)
            alignmentFormat = "<RIGHT>";
        else if (align == CENTER)
            alignmentFormat = "<CENTER>";
    }


    /**
     * stick the text content in the name String
     * override this to just render the name (i.e. the text content)
     * without start or end tags
     */
    public String render() {
        if (lineFormat == null && alignmentFormat == null) return getName();

        if (lineFormat == null) lineFormat = "";
        if (alignmentFormat == null) alignmentFormat = "";

        return lineFormat + alignmentFormat + getName();
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * base class for all tags in the HDML tag hierarchy
 */
public abstract class HDMLTag extends Tag {

    /**
     * Create a new HDML tag that may optionally close itself
     * @param name for this card
     * @param closingTag if true then this tag closes itself
     */
    public HDMLTag(String name, boolean closingTag) {
        super(name, closingTag);
    }

    /**
     * Create a HDML tag that closes itself
     * @param name for this card
     */
    public HDMLTag(String name) {
        super(name, false);
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * Represents an entire deck of an hdml document.
 * <p>
 * Contains the topmost <HDML> tag, any cards, AND any actions, and the </HDML>
 */
public class HDMLTagDocument extends TagDocument {

    /**
     * create a new HDMLTagDocument with markable and public set to true
     */
    public HDMLTagDocument() {
        this(true, true);
    }


    /**
     * create a new HDMLTagDocument
     * with settings for MARKABLE and PUBLIC
     * @param markable if true then this document is markable
     * @param pub then this document is public
     */
    public HDMLTagDocument(boolean markable, boolean pub) {
        super("HDML", "text/x-hdml");
        getRoot().addAttribute( new Attribute("VERSION", "3.0", false) );

        setPublic(pub);
        setMarkable(markable);
    }

    /**
     * set the time to live option
     * @param seconds number of seconds that the deck will be cached by the user agent
     * after reception
     */
    public void setTimeToLive(int seconds) {
        getRoot().addAttribute( new Attribute("TTL", String.valueOf(seconds), false) );
    }


    /**
     * set the public option
     * @param b indicates whether deck access control has been enabled for this deck
     */
    public void setPublic(boolean b) {
        String val = ((b) ? "true" : "false");
        getRoot().addAttribute( new Attribute("PUBLIC", val, false) );
    }


    /**
     * set the access domain of this document
     * @param url String domain name, i.e. something.com the access domain is suffix-matched ↙
         against the domain
     * name of a referring url
     */
    public void setAccessDomain(String url) {

        if (getRoot().getAttribute("PUBLIC") == null)
            throw new InvalidHDMLException("The public option must be set to true before an ↙
                Access Domain can be set");

        getRoot().addAttribute( new Attribute("ACCESSDOMAIN", url) );
    }
    /**
     * set the access path of this document
     * @param path String can be a relative url, i.e. /thinair/docs
     */
    public void setAccessPath(String path) {

        if (getRoot().getAttribute("PUBLIC") == null)
            throw new InvalidHDMLException("The public option must be set to true before an ↙
                Access Path can be set");

        getRoot().addAttribute( new Attribute("ACCESSPATH", path) );
    }


    /**
     * set the markable option
     * @param b specifies whether the cards in this deck can be bookmarked or not
     */
    public void setMarkable(boolean b) {
        if ( b == true && (getRoot().getAttribute("PUBLIC") == null))
            throw new InvalidHDMLException("The public option must be set to true before an ↙
                Markable can be set to true");

        String val = ((b) ? "true"  : "false");
        getRoot().addAttribute( new Attribute("MARKABLE", val, false) );
    }


    /**
     * add an action to the TaggedDocument
     * actions added at this level remain in effect for the life of the deck
     * when an action is specified for a card it overrides any deck actions of that type
     * while the card is visible
     * @param action an Action bound to some button on the device
     */
    public void addAction(Action action) {
        try {
            getRoot().addChild(action);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

    /**
     * add a Card to this Tagged Document
     * @param c Card to add to the document
     */
    public void addCard(Card c) {
        try {
            getRoot().addChild(c);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }


    /**
     * when you go to render - there must be >= 1 card added to the deck
     */
    public String render() {

        if (getRoot().getChildren().size() > 0)
            return super.render();
        else
            throw new InvalidHDMLException("One or more cards must be added to a deck before ↙
                rendering");
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * An image tag, supported only by some HDML-rendering device
 */
public class ImageTag extends HDMLTag {

    /**
     * Create a new HDML Image
     * @param url location of the device
     * @param altText to appear if no image can be displayed
     */
    public ImageTag(String url, String altText) {
        super("IMG", false);
        addAttribute( new Attribute("SRC", url) );
        addAttribute( new Attribute("ALT", altText) );
    }

    /** Create an imagae tag with no url or alt text */
    public ImageTag() {
        super("IMG", false);
    }


    /**
     * @param iconName name for the icon
     */
    public void setIcon(String iconName) {
        addAttribute(new Attribute("ICON",iconName));
    }

    /**
     * @param altText to be displayed if the device does not support images
     */
    public void setAltText(String altText) {
        addAttribute( new Attribute("ALT", altText) );
    }


    /**
     * set the NAME option
     * alternative internal graphic representation for the image
     * if an image by name exists it will be used - otherwise one will be downloaded
     *  from the src URL
     * this allows user-agents to provide an internal set of generic images identified by
        name
     * @param name for this image tag
     */
    public void setName(String name) {
        addAttribute( new Attribute("NAME", name) );
    }

    /**
     * @return String url for this image
     */
    public String getSrc() { return (String) getAttribute("SRC").getValue(); }

} // end
```

```
package com.thinairapps.tag.hdml;

/**
 * represents an invalid tag or document layout
 */
public class InvalidHDMLException extends RuntimeException {

    /**
     * Create InvalidHDMLException
     * @param message carried inside the exception
     */
    public InvalidHDMLException(String message) {
        super(message);
    }

    /** create an InvalidHDMLException with no message */
    public InvalidHDMLException() {
        super("invalid HDML used");
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * The NODISPLAY card DOES NOT give information for the user to read
 * <p>
 * It immeditately executes its ACCEPT or PREV action -- all other
 * actions are ignored - this implementation will throw an exception
 * if any other action type is entered
 */
public abstract class NoDisplayCard extends Card {

    /**
     * create a display card
     */
    public NoDisplayCard() {
        super("NODISPLAY", true);
    }

    /**
     * add an action to this Card
     * when an action is specified for a card it overrides any deck actions of that type
     * while the card is visible
     * @param action to bind to some button on the device
     */
    public void addAction(Action action) {

        Action.Type type = action.getType();
        if (type == Action.ACCEPT || type == Action.PREV)
            ; // ok
        else
            throw new InvalidHDMLException("Only Accept and Prev actions are valid for a
                NODISPLAY Card");

        try {
            super.addChild(action);
        } catch (Exception e) {
            throw new InvalidHDMLException(e.getMessage());
        }
    }

} // end
```

```
package com.thinairapps.tag.hdml;

import com.thinairapps.tag.*;

/**
 * Represents the ENTRY card
 * <p>
 * let users input an optionally formatted character string - the display content
 * is shown to the user, followed by an area to input characters
 */
public class PasswordEntryCard extends EntryCard {

    /** create a blank PasswordEntryCard with no destination */
    public PasswordEntryCard() {
        super();
        setNoEcho(true);
    }


    /**
     * create a password entry card
     * upon completion of password the next card in the deck
     * is identified by dest
     * @param dest maps dest to $(username)
     */
    public PasswordEntryCard(String dest) {
        this(dest, "$(username)");
    }


    /**
     * create a password entry card
     * upon completion of password the next card in the deck
     * is identified by dest
     *
     * @param dest String url to go to when the password is entered
     * @param username name of user of this card
     */
    public PasswordEntryCard(String dest, String username) {
        super();

        setName("passwordCard");
        setKey("password");
        setTitle("Enter Password:");

        ActionTask task = new ActionTask(ActionTask.GOSUB);
        task.setVar("password", "$(password)");
        task.setDest(dest);
        addAction( new Action(Action.ACCEPT, task, "Next") );

        setNoEcho(true);
        setEmptyOK(false);

        if (username != null) addText( new FormattedLine("User: "+username) );
        addText( new FormattedLine("Password:") );
    }



    /**
     * create a password entry card
     * upon completion of password the next card in the deck
     * is identified by dest
     *
     * @param dest String url to go to when the password is entered
     * @param username name of user of this card
     */
    public PasswordEntryCard(String dest, String key, String username) {
        super();
        setName("passwordCard");
        setKey(key);
        setTitle("Enter Password:");

        ActionTask task = new ActionTask(ActionTask.GOSUB);
        task.setVar("password", "$(password)");
        task.setDest(dest);
        addAction( new Action(Action.ACCEPT, task, "Next") );

        setNoEcho(true);
        setEmptyOK(false);

        if (username != null) addText( new FormattedLine("User: "+username) );
        addText( new FormattedLine("Password:") );
    }

} // end
```

```
package com.thinairapps.tag.hdml;

/**
 * This class allows you to insert an arbitrary String
 * into a HDML Deck.
 */
public class Text extends HDMLTag {

    /**
     * Create a new Text Object with the given String
     * @param text String text for this tag
     */
    public Text(String text) {
        super(text,false);
    }

    /**
     * @return String you used to define this Text Object
     */
    public String getTextOnly() {
        return getName();
    }

    /**
     * @return Sting HDML markup for this object
     */
    public String render() {
        return getName();
    }

} // end
```

```
/**
 * @(#)WAPDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps abstraction for a WAP device
 */
public class WAPDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 2962217947595361097L;

    /**
     * Constructs a new <code>WAPDevice</code> instance of the type represented by profile.
     * @param profile <code>WAPDeviceProfile</code> prototype for this device instance
     */
    WAPDevice (WAPDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType () {
        return WAPDeviceProfile.WML_USER_CONTENT_TYPE;
    }

} // end
```

```
/**
 * @(#)WAPDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>
   factory for WAP devices
 */
public class WAPDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 3341161013503207 63L;


    /** name of this device profile */
    public static final String NAME = "TA_WAP";

    /** content type accepted by all devices matching this profile */
    static final String WML_USER_CONTENT_TYPE  = "text/vnd.wap.wml";

    /** alternate content type accepted by all devices matching this profile */
    static final String WML_USER_CONTENT_TYPE2 = "text/x-wap.wml";

    /** alternate mime type indicating wml compliance */
    static final String WML_USER_CONTENT_TYPE3 = "application/vnd.wap.wmlc";

    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile
        creates.
     *
     * @return <code>Class</code> of the <code>WAPDevice</code> instances that this profile
        generates.
     */
    public Class getDeviceClass () { return new WAPDevice (this).getClass (); }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>WAPDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>WAPDevice
        </code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check Accept header
            HttpServletRequest request = (HttpServletRequest)req;
            String accept = request.getHeader("Accept");

            return ( (accept != null ) && (accept.indexOf ("wap.wml") >= 0));
        }
        return false;
    }


    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
        requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        WAPDevice device = new WAPDevice (this);

        // need to check for presence of x-up-subno, if not there, do not set GUID
        String tempGUID = request.getHeader("x-up-subno");
        if (tempGUID != null) device.setGUID( NAME + ":" + tempGUID );

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        String tempUserAgent = request.getHeader("User-Agent");
        device.setUserAgent( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // COOKIES
        device.setCookies (request.getCookies ());


        return device;
    }



    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
        NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        WAPDevice device = new WAPDevice(this);

        device.setGUID(guid);

        // inintialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);

        return device;
    }

} // end
```

```
/**
 * @(#)UPWAPDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>      ↙
    factory for up.com WAP phones
 */
public class UPWAPDeviceProfile extends WAPDeviceProfile {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -2773505273328761059L;



    /** name of this device profile */
    public static final String NAME = "TA_UP_WAP";
    /** user-agent transmitted with every request coming from devices matching this profile  ↙
        */
    static final String USER_AGENT = "UP";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>UPWAPDevice</code> instances that this profile↙
        generates.
     */
    public Class getDeviceClass() {
        return new UPWAPDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>   ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>UPWAPDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>         ↙
        UPWAPDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice(ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String userAgent = request.getHeader("User-Agent");

            return ( (userAgent != null ) && (userAgent.indexOf(USER_AGENT) >= 0));
        }

        return false;
    }
    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest)req;
        UPWAPDevice device = new UPWAPDevice (this);

        // declare temp values for device properties, to check for null
        String  tempGUID,
                tempUserAgent,
                tempLanguage,
                tempFax,
                tempCharset,
                tempHost,
                tempSmartDialing,
                tempScreenDepth,
                tempColor,
                tempAlert,
                tempPDU,
                tempSoftKeys,
                tempScreenChars,
                tempPixels;


        // initialize temps
        tempGUID = request.getHeader("x-up-subno");
        tempUserAgent = request.getHeader ("User-Agent");
        tempLanguage = request.getHeader("Accept-Language");
        tempFax = request.getHeader("x-upfax-accepts");
        tempCharset = request.getHeader("accept-charset");
        tempHost = request.getHeader("Host");
        tempSmartDialing = request.getHeader ("x-up-devcap-smartdialing");
        tempScreenDepth =  request.getHeader ("x-up-devcap-screendepth");
        tempAlert = request.getHeader("x-up-devcap-immed-alert");
        tempColor = request.getHeader ("x-up-devcap-iscolor");
        tempPDU = request.getHeader ("x-up-devcap-max-pdu");
        tempSoftKeys = request.getHeader ("x-up-devcap-numsoftkeys");
        tempScreenChars = request.getHeader ("x-up-devcap-screenchars");
        tempPixels = request.getHeader ("x-up-devcap-screenpixels");



        // **NEED TO VERIFY IF THIS IS ALWAYS UNIQUE** - gordogre 10/18/2000
        if (tempGUID != null) device.setGUID( UPWAPDeviceProfile.NAME + ":" + tempGUID );

        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // ACCEPT-LANGUAGE
        device.language = (tempLanguage == null) ? STRING_NO_VALUE : tempLanguage;

        // ACCEPT-FAX
        device.acceptFax = (tempFax == null) ? STRING_NO_VALUE : tempFax;

        // ACCEPT-CHARSET
        device.acceptCharset = (tempCharset == null) ? STRING_NO_VALUE : tempCharset;

        // HOST
        device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // SMART DIALING
        if (tempSmartDialing != null)
            device.smartDialing = tempSmartDialing.equals ("1") ? true : false;

        // SCREEN DEPTH
        if (tempScreenDepth != null)
        {
            try
                {
                    device.screenDepth = Integer.parseInt(tempScreenDepth);
                }

            catch (Exception e)
                {
                    device.screenDepth = INT_NO_VALUE;
                }
        }
        else
        {
            device.screenDepth = INT_NO_VALUE;
        }


        // IS COLOR
        if (tempColor != null)
            device.isColor = tempColor.equals ("1") ? true : false;

        // IMMEDIATE ALERT
        if (tempAlert != null)
            device.immediateAlert = tempAlert.equals ("1") ? true : false;

        // MAX PDU
        device.maxPDU = (tempPDU == null) ? STRING_NO_VALUE : tempPDU;

        // NUMBER OF SOFT KEYS
        if (tempSoftKeys != null)
        {
            try
                {
                    device.softKeys = Integer.parseInt (tempSoftKeys);
                }
            catch (Exception e)
                {
                    device.softKeys = INT_NO_VALUE;
                }
        }
        else
        {
            device.softKeys = INT_NO_VALUE;
        }

        // SCREEN CHARACTERS
        device.screenChars = (tempScreenChars == null) ? STRING_NO_VALUE : tempScreenChars;

        // SCREEN PIXELS
        if (tempPixels != null)
            {

            try
                {
                    device.pixelWidth = Integer.parseInt (tempPixels.substring (0, tempPixels↙
                        .indexOf (",")));
                    device.pixelHeight = Integer.parseInt (tempPixels.substring (tempPixels. ↙
                        indexOf (",") + 1));
                }
            catch (Exception e)
                {
                    // This shouldn't kill everything, so we'll quietly catch it
                    device.pixelHeight = INT_NO_VALUE;
                    device.pixelWidth = INT_NO_VALUE;
                }
            }
        else
        {
            device.pixelHeight = INT_NO_VALUE;
            device.pixelWidth = INT_NO_VALUE;
        }


        return device;
    }



    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        UPWAPDevice device = new UPWAPDevice(this);

        device.setGUID( guid );

        // inintialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.language = STRING_NO_VALUE;
        device.acceptFax = STRING_NO_VALUE;
        device.acceptCharset = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.screenDepth = INT_NO_VALUE;
        device.maxPDU = STRING_NO_VALUE;
        device.softKeys = INT_NO_VALUE;
        device.screenChars = STRING_NO_VALUE;
        device.pixelWidth = INT_NO_VALUE;
        device.pixelHeight =INT_NO_VALUE;

        return device;
    }

} // end


/*
Content-Type: application/x-www-form-urlencoded
Accept-Charset: ISO-8859-1, UTF-8, *
x-up-subno: profix_LT-CT-6220
x-upfax-accepts: none
x-up-devcap-charset: ISO-8859-1
Accept: application/x-hdmlc, application/x-up-alert, application/x-up-cacheop, application/
    x-up-device, application/x-up-digestentry, application/vnd.wap.wml, text/x-wap.wml, text/
    vnd.wap.wml, application/vnd.wap.wmlscript, text/vnd.wap.wmlscript, application/vnd.
    uplanet.channel, application/vnd.uplanet.list, text/x-hdml, text/plain, text/html, image/
    vnd.wap.wbmp, image/bmp, application/remote-printing text/x-hdml;version=3.1, text/x-hdml
    ;version=3.0, text/x-hdml;version=2.0, image/bmp, text/html
User-Agent: ALAV UP/4.0.10 UP.Browser/4.0.10-XXXX UP.Link/4.1.HTTP-DIRECT
*/
```

```java
/**
 * @(#)UPWAPDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for up.com WAP phones
 */
public class UPWAPDevice extends WAPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -9047029144031715525L;

    protected boolean smartDialing;
    protected boolean isColor;
    protected boolean immediateAlert;
    protected int     softKeys;
    protected int     screenDepth;
    protected int     pixelWidth;
    protected int     pixelHeight;
    protected String  language;
    protected String  maxPDU;
    protected String  screenChars;
    protected String  host;
    protected String  acceptFax;
    protected String  acceptCharset;


    /**
     * Constructs a new <code>UPWAPDevice</code> instance of the type represented by profile.
     *
     * @param profile <code>UPWAPDeviceProfile</code> prototype for this device instance
     */
    UPWAPDevice (UPWAPDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves a value indicating whehter the device accepts UP faxes.
     * <p>
     * Value is typically "1" if device does accept faxes, "0" if it does not.
     *
     * @return <code>String</code> accepts UP-Fax
     */
    public String getAcceptFax() { return acceptFax; }


    /**
     * Retrieves the character encoding set supported by the device.
     *
     * @return <code>String</code> character set supported by device    */
    public String getAcceptCharset() { return acceptCharset; }


    /**
     * Retrieves the language locale supported by the device.
     *
     * @return <code>String</code> language locale, if specified
     */
    public String getLanguage() { return language; }

    /**
     * Retreives flag indicating if smart dialing is enabled on this device.
     *
     * @return <code>boolean</code> true if smart dialing enabled, false otherwise
     */
    public boolean isSmartDialing() { return smartDialing; }

    /**
     * Retrieves the screen depth of this device.
     *
     * @return <code>int</code> screen bit depth
     */
    public int getScreenDepth() { return screenDepth; }

    /**
     * Retreives flag indicating whether this is a color device.
     *
     * @return <code>boolean</code> true if device is color, false otherwise
     */
    public boolean isColor() { return isColor; }

    /**
     * Retreives flag indicating if immediate alert is enabled on this device.
     *
     * @return <code>boolean</code> true if immediate alert enabled, false otherwise
     */
    public boolean immediateAlert() { return immediateAlert; }

    /**
     * Retrieves the maximum size of PDU's this device supports.
     *
     * @return <code>String</code> max PDU
     */
    public String getMaxPDU() { return maxPDU; }

    /**
     * Retrieves the number of soft keys on this device.
     *
     *  @return <code>int</code> number of soft keys
     */
    public int numSoftKeys() { return softKeys; }

    /**
     * Retrieves the dimension of the devices screen in terms of characters.
     * The format for this value is: "H,V", where H is horizontal characters (rows) and V is ↙
       vertical characters (columns).
     * @return <code>String</code> screen characters
     */
    public String getScreenChars() { return screenChars; }

    /**
     * Retreives the width of the device's screen in pixels.
     *
     * @return <code>int</code> screen pixel width
     */
    public int getPixelWidth() { return pixelWidth; }

    /**
     * Retrieves the height of the device's screen in pixels.
     *
     * @return <code>int</code> screen pixel height
     */
    public int getPixelHeight() { return pixelHeight; }

} // end
```

```
/**
 * @(#)TellMeDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>
   factory for HDML devices
 */
public class TellMeDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 2489998979065444352L;

    /** name of this device profile */
    public static final String NAME = "TA_TELLME";
    /** content type accepted by all devices matching this profile */
    protected static final String CONTENT_TYPE = "text/xml";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile
       creates.
     *
     * @return <code>Class</code> of the <code>TellMeDevice</code> instances that this
       profile generates.
     */
    public Class getDeviceClass() {
        return new TellMeDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>
       DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>TellMeDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>
       TellMeDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {


        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String userAgent = request.getHeader("User-Agent");

            return ( (userAgent != null ) &&
                     (userAgent.equals("Tellme/1.0 (I; en-US)") || userAgent.equals
                        ("libwww-perl/5.47")) );
        }

        return false;
    }
    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the ↙
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        TellMeDevice device = new TellMeDevice (this);

        // no known value to use for GUID, so do not set

        // USER-AGENT
        String tempUserAgent = request.getHeader("User-Agent");
        device.setUserAgent( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // COOKIES
        device.setCookies(request.getCookies());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate ↙
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        TellMeDevice device = new TellMeDevice(this);

        // TellMeDevice device does not have GUID, so do not set

        // inintialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);

        return device;
    }

} // end
```

```
/**
 * @(#)TellMeDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for the TellMe browser
 */
public class TellMeDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -3692689441219031923L;

    /**
     * Constructs a new <code>TellMeDevice</code> instance of the type represented by profile↙
     .
     * @param profile <code>TellMeDeviceProfile</code> prototype for this device instance
     */
    TellMeDevice (TellMeDeviceProfile profile) {
        super(profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType() {
        return TellMeDeviceProfile.CONTENT_TYPE;
    }

} // end
```

```
/**
 * @(#)PocketIEDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>     ↙
    factory for Pocket IE devices
 */
public class PocketIEDeviceProfile extends HTTPDeviceProfile {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 4004923160351708628L;

    /** name of this device profile */
    public static final String NAME = "TA_POCKETIE";

    /** content type accepted by all devices matching this profile */
    public static final String CONTENT_TYPE = "text/html";

    /** user agent transmitted with every request from this device */
    public static final String USER_AGENT = "Mozilla/2.0 (compatible; MSIE 3.02; Windows     ↙
        CE)";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>PocketIEDevice</code> instances that this     ↙
        profile generates.
     */
    public Class getDeviceClass() {
        return new PocketIEDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>   ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>PocketIEDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>         ↙
        PocketIEDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String userAgent = request.getHeader("User-Agent");

            return ( (userAgent != null ) && (userAgent.indexOf("Windows CE") >= 0));
        }

        return false;
    }



    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        PocketIEDevice device = new PocketIEDevice (this);

        // no known value to use for GUID, so do not set

        // declare temp values for device properties, to check for null
        String  tempUserAgent,
                tempHost,
                tempOS,
                tempColor,
                tempScreenPixels;

        // initialize temps
        tempUserAgent = request.getHeader ("User-Agent");
        tempHost = request.getHeader("Host");
        tempOS = request.getHeader("UA-OS");
        tempColor = request.getHeader("UA-color");
        tempScreenPixels = request.getHeader("UA-pixels");


        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // HOST
        device.m_host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // OS
        device.m_os = (tempOS == null) ? STRING_NO_VALUE : tempOS;

        // COLOR
        device.m_color = (tempColor == null) ? STRING_NO_VALUE : tempColor;

        // SCREEN PIXELS
        device.m_screen = (tempScreenPixels == null) ? STRING_NO_VALUE : tempScreenPixels;


        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        PocketIEDevice device = new PocketIEDevice(this);

        // PocketIE device does not have GUID, so do not set

        // inintialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.m_host = STRING_NO_VALUE;
        device.m_os = STRING_NO_VALUE;
        device.m_color = STRING_NO_VALUE;
        device.m_screen = STRING_NO_VALUE;

        return device;
    }

} // end

//Accept: */*
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: UA-OS: Windows CE ↙
    (POCKET PC) - Version 3.0
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: UA-color: color32
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: UA-pixels: 240x320
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: Referer: http://10↙
    .1.1.61/taaps/html
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: UA-Language:      ↙
    JavaScript
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: Accept-Encoding:  ↙
    gzip, deflate
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: User-Agent:       ↙
    Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; 240x320)
//05/17/2000 02:14:14 PM: BasicHTTPServletRequest.parseRequest: http line: Host: 10.1.1.61


/*
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: Host: 10.1.1.61
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: UA-OS: Windows CE   ↙
    () - Version 2.11
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: UA-color:
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: UA-pixels: 640x480
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: UA-CPU: Unknown     ↙
    (586)
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: Cookie:
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: Pragma: no-cache
05/17/2000 07:45:34 PM: BasicHTTPServletRequest.parseRequest: http line: Accept:
*/
```

```
/**
 * @(#)PocketIEDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps abstraction for devices supporting Microsoft Pocket IE
 */
public class PocketIEDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -3014994380237056646L;

    protected String m_os, m_color, m_screen, m_host;


    /**
     * Constructs a new <code>PocketIEDevice</code> instance of the type represented by
       profile.
     * @param profile <code>PocketIEDeviceProfile</code> prototype for this device instance
     */
    PocketIEDevice (PocketIEDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType () { return PocketIEDeviceProfile.CONTENT_TYPE; }

    /**
     * Retrieves a description of the operating system running on the device.
     *
     * @return <code>String</code> OS of this device */
    public String getOS() { return m_os; }

   /**
     * Retrieves the color depth of the screen for the device.
     *
     * @return <code>String</code> color depth of device
     */
    public String getColorDepth() { return m_color; }

    /**
     * Retrieves the screen size of the device.
     * <p>
     * The returned value will be in the format "HxW", where H is the height of the screen
     * in pixels, and W is the width of the screen in pixels.
     *
     * @return <code>String</code> screen size of the device.
     */
    public String getScreenSize() { return m_screen; }

    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return m_host; }

} // end
```

```
/**
 * @(#)PalmVIIDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>      ↙
    factory for Palm VII devices
 */
public class PalmVIIDeviceProfile extends HTTPDeviceProfile {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 6054987558599139231L;

    /** name of this device profile */
    public static final String NAME = "TA_PALM_VII";

    /** user-agent transmitted with every request coming from devices matching this profile  ↙
        */
    // public static final String USER_AGENT   = "Mozilla/2.0 (compatible; Elaine/1.0)";

    /** content type accepted by all devices matching this profile */
    public static final String CONTENT_TYPE = "text/html";

    // seems like Elaine sends user-agent as a lower case string
    private static final String userAgentKey = "user-agent";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>PalmVIIDevice</code> instances that this      ↙
        profile generates.
     */
    public Class getDeviceClass() {
        return new PalmVIIDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>    ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>PalmVIIDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>         ↙
        PalmVIIDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String userAgent = request.getHeader("User-Agent");

            // sgross 12/12/2000:
```

```
            // This change ensures that Elaine servers which are ABOVE 1.*
            // will not be recognized by this profile
            // Reports are that the content-encodings for Elaine/2.* servers will
            // be incompatible with our software
            return ( (userAgent != null ) && (userAgent.indexOf("Elaine/1") >= 0));
        }

        return false;
    }


    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        PalmVIIDevice device = new PalmVIIDevice (this);


        // this may not be present if not originating from our Palm Secure Client - gordogre
        String guid = request.getParameter ("d");
        if (guid == null)
            guid = request.getParameter ("did");
        if (guid == null)
            guid = request.getParameter ("deviceid");

        // if we do not find a suitable guid, we should not set device GUID - gordogre
        if (guid != null) device.setGUID(NAME + ":" + guid);

        // declare temp values for device properties, to check for null
        String  tempUserAgent,
                tempHost,
                tempVia,
                tempConnection,
                tempForwardedFor,
                tempElaineVersion;

        // initialize temps
        tempUserAgent = request.getHeader ("User-Agent");
        tempHost = request.getHeader("Host");
        tempVia = request.getHeader("Via");
        tempConnection = request.getHeader("Connection");
        tempForwardedFor = request.getHeader("X-Forwarded-For");
        tempElaineVersion = parseElaineVersion(tempUserAgent);


        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // VIA
        device.via = (tempVia == null) ? STRING_NO_VALUE : tempVia;

        // HOST
        device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // CONNECTION
        device.connectionType = (tempConnection == null) ? STRING_NO_VALUE : tempConnection;

        // FORWARDED-FOR
        device.forwardAddress = (tempForwardedFor == null) ? STRING_NO_VALUE :          ↙
            tempForwardedFor;

        // ELAINE VERSION
        device.elaineVersion = (tempElaineVersion == null) ? STRING_NO_VALUE :          ↙
            tempElaineVersion;


        return device;
    }



    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate     ↙
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        PalmVIIDevice device = new PalmVIIDevice(this);

        device.setGUID( guid );

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.via = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.connectionType = STRING_NO_VALUE;
        device.forwardAddress = STRING_NO_VALUE;
        device.elaineVersion = STRING_NO_VALUE;

        return device;
    }


    /**
     * Parses User-Agent header to retrieve version of Elaine browser
     *
     * @param <code>String</code> User-Agent header from request
     *
     * @return <code>String</code> Elaine version making request, or null if the Elaine     ↙
       version cannot be obtained
     */
    private String parseElaineVersion(String ua)
    {
        String temp;
        int index;

        // first check for null string
        if (ua == null)
        {
            return null;
        }
        else
        {
            // obtain index of string preceding version number; i.e. "Mozilla/2.0 (compatible↙
                ; Elaine/1.0)"
            index = ua.indexOf("Elaine/");
            // now obtain substring begining with versio number
            temp = ua.substring(index + 7);

            // obtain ending index of version number
            index = temp.indexOf(")");

            if (index > 0)
            {   // return substring containing version number only
                return temp.substring(0,index);
            }
            else
            {   // if version number not present, return null
                return null;
            }
        }
    }

} // end
```

```
/**
 * @(#)PalmVIIDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for the Palm VII PDA
 */
public class PalmVIIDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 4619797748056876641L;

    protected String forwardAddress, host, via, connectionType, elaineVersion;

    /**
     * Constructs a new <code>PalmVIIDevice</code> instance of the type represented by
        profile.
     * @param profile <code>PalmVIIDeviceProfile</code> prototype for this device instance
     */
    PalmVIIDevice (PalmVIIDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType() {
        return PalmVIIDeviceProfile.CONTENT_TYPE;
    }

    /**
     * Retrieves the connection type for the request.
     * <p>
     * Typical values are "Keep-Alive" or "Close".
     *
     * @return <code>String</code> connection type for the request
     */
    public String getConnectionType() { return connectionType; }

    /**
     * Retrieves the IP address of the device.
     * This is not necessarily a unique client IP address!
     *
     * @return String IP address of device
     */
    public String getForwardAddress() { return forwardAddress; }

    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return host; }

    /**
     * Retrieves the domain of the device's gateway.
     *
     * @return <code>String</code> domain of request origin (gateway domain)
     */
    public String getGateway() { return via; }

    /**
     * Retrieves the version of the Elaine browser running on the device.
     *
     * @return <code>String</code> version of Elaine browser making request
     */
    public String getElaineVersion() { return elaineVersion; }

} // end
```

```
/**
 * @(#)OmniSkyDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>   ↙
   factory for OmniSky devices
 */
public class OmniSkyDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 7531455097521934005L;


    /** name of this device profile */
    public static final String NAME = "TA_OMNISKY";
    /** user-agent transmitted with every request coming from devices matching this profile  ↙
        */
    static final String CONTENT_TYPE = "text/html";
    /** content type accepted by all devices matching this profile */
    static final String USER_AGENT = "Mozilla/2.0 (compatible; Elaine/1.0)";

    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>OmniSkyDevice</code> instances that this     ↙
        profile generates.
     */
    public Class getDeviceClass() { return new OmniSkyDevice (this).getClass(); }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>   ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>OmniSkyDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>        ↙
        OmniSkyDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String userAgent = request.getHeader("User-Agent");

            // must also check "via" header, which will contain "OMMD" for OmniSky gateways
            String tempVia = request.getHeader("Via");

            return ( (userAgent != null ) &&
                     (tempVia != null) &&
                     (userAgent.indexOf("Elaine") >= 0) &&
                     (tempVia.indexOf("OMMD") >= 0) );
        }

        return false;
    }

    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the    ↙
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest)req;
        OmniSkyDevice device = new OmniSkyDevice (this);

        // this may not be present if not originating from our Palm Secure Client - gordogre
        String guid = request.getParameter ("d");
        if (guid == null)
            guid = request.getParameter ("did");
        if (guid == null)
            guid = request.getParameter ("deviceid");

        // if we do not find a suitable guid, we should not set device GUID - gordogre
        if (guid != null) device.setGUID(NAME + ":" + guid);


        // declare temp values for device properties, to check for null
        String  tempUserAgent,
                tempHost,
                tempVia,
                tempElaineVersion;

        // initialize temps
        tempUserAgent = request.getHeader ("User-Agent");
        tempHost = request.getHeader("Host");
        tempVia = request.getHeader("Via");
        tempElaineVersion = parseElaineVersion(tempUserAgent);

        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // VIA
        device.via = (tempVia == null) ? STRING_NO_VALUE : tempVia;

        // HOST
        device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // ELAINE VERSION
        device.elaineVersion = (tempElaineVersion == null) ? STRING_NO_VALUE :    ↙
            tempElaineVersion;


        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate     ↙
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        OmniSkyDevice device = new OmniSkyDevice(this);

        device.setGUID( guid );

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.via = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.elaineVersion = STRING_NO_VALUE;

        return device;
    }


    /**
     * Parses User-Agent header to retrieve version of Elaine browser
     *
     * @param <code>String</code> User-Agent header from request
     *
     * @return <code>String</code> Elaine version making request, or null if the Elaine     ↙
       version cannot be obtained
     */
    private String parseElaineVersion(String ua)
    {
        String temp;
        int index;

        // first check for null string
        if (ua == null)
        {
            return null;
        }
        else
        {
            // obtain index of string preceding version number; i.e. "Mozilla/2.0 (compatible↙
                ; Elaine/1.0)"
            index = ua.indexOf("Elaine/");

            // now obtain substring begining with versio number
            temp = ua.substring(index + 7);

            // obtain ending index of version number
            index = temp.indexOf(")");

            if (index > 0)
            {   // return substring containing version number only
                return temp.substring(0,index);
            }
            else
            {   // if version number not present, return null
                return null;
            }
        }
    }

} // end
```

```
/**
 * @(#)OmniSkyDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for OmniSky-enabled devices
 */
public class OmniSkyDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -8857853771839964701L;

    protected String host, via, elaineVersion;

    /**
     * Constructs a new <code>OmniSkyDevice</code> instance of the type represented by
        profile.
     * @param profile <code>OmniSkyDeviceProfile</code> prototype for this device instance
     */
    OmniSkyDevice (OmniSkyDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType () {
        return OmniSkyDeviceProfile.CONTENT_TYPE;
    }

    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return host; }

    /**
     * Retrieves the domain of the device's gateway.
     *
     * @return <code>String</code> domain of request origin (gateway domain)
     */
    public String getGateway() { return via; }

    /**
     * Retrieves the version of the Elaine browser running on the device.
     *
     * @return <code>String</code> version of Elaine browser making request
     */
    public String getElaineVersion() { return elaineVersion; }

} // end
```

```
/**
 * @(#)NokiaWAPDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>    ↙
    factory for NOKIA WAP phones
 */
public class NokiaWAPDeviceProfile extends WAPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -7297723379847931620L;


    /** name of this device profile */
    public static final String NAME = "TA_NOKIA_WAP";

    // FIXME FIXME FIXME: set the proper user agent
    /** user-agent transmitted with every request coming from devices matching this profile  ↙
        */
    static final String USER_AGENT = "nokia-user-agent";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>NokiaWAPDevice</code> instances that this    ↙
        profile generates.
     */
    public Class getDeviceClass() {
        return new NokiaWAPDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>  ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>NokiaWAPDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>        ↙
        NokiaWAPDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

       if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String userAgent = request.getHeader("User-Agent");

            return ( (userAgent != null ) && (userAgent.indexOf("Nokia") >= 0));
        }

        return false;
    }
```

```
/**
 * Create a <code>Device</code> instance with the same properties as the actual
 * physical device that generated this servlet request.
 *
 * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
 *
 * @return <code>Device</code> a device instance with properties set to describe the
 *   requesting device.
 */
public Device createDeviceFromRequest (ServletRequest req) {

    HttpServletRequest request = (HttpServletRequest)req;
    NokiaWAPDevice device = new NokiaWAPDevice (this);

    // cannot verify that x-network-info will be unique, so do
    // not set GUID - gordogre

    // declare temp values for device properties, to check for null
    String  tempUserAgent,
            tempEncoding,
            tempHost,
            tempVia,
            tempLanguage,
            tempCharset,
            tempNetworkInfo;

    // initialize temps
    tempUserAgent = request.getHeader("User-Agent");
    tempEncoding = request.getHeader ("Accept-Encoding");
    tempHost = request.getHeader("Host");
    tempVia = request.getHeader("Via");
    tempLanguage = request.getHeader("Accept-Language");
    tempCharset = request.getHeader("Accept-Charset");
    tempNetworkInfo = request.getHeader("x-network-info");


    // COOKIES
    device.setCookies (request.getCookies ());

    // ACCEPT
    device.setAccept(request.getHeader("Accept"));

    // USER-AGENT
    device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

    // ACCEPT-LANGUAGE
    device.acceptLanguage = (tempLanguage == null) ? STRING_NO_VALUE : tempLanguage;

    // HOST
    device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

    // ACCEPT-ENCODING
    device.acceptEncoding = (tempEncoding == null) ? STRING_NO_VALUE : tempEncoding;

    // VIA
    device.via = (tempVia == null) ? STRING_NO_VALUE : tempVia;

    // ACCEPT-CHARSET
    device.acceptCharset = (tempCharset == null) ? STRING_NO_VALUE : tempCharset;

    // NETWORK-INFO
    device.networkInfo = (tempNetworkInfo == null) ? STRING_NO_VALUE : tempNetworkInfo;


    return device;
}
```

```
    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
        NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        NokiaWAPDevice device = new NokiaWAPDevice(this);

        // NokiaWAPDevices have no GUID, so do not set

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.acceptLanguage = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.acceptEncoding = STRING_NO_VALUE;
        device.via = STRING_NO_VALUE;
        device.acceptCharset = STRING_NO_VALUE;
        device.networkInfo = STRING_NO_VALUE;

        return device;
    }

} // end
```

```
/**
 * @(#)NokiaWAPDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps' abstraction for the Nokia family of WAP phones
 */
public class NokiaWAPDevice extends WAPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 3955919879913874076L;

    protected String host, acceptLanguage, via, acceptEncoding,
                     acceptCharset, networkInfo;

    /**
     * Constructs a new <code>NokiaWAPDevice</code> instance of the type represented by ↙
       profile.
     * @param profile <code>NokiaWAPDeviceProfile</code> prototype for this device instance
     */
    NokiaWAPDevice (NokiaWAPDeviceProfile profile) {
        super (profile);
    }


    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return host; }


    /**
     * Retrieves the language locale supported by the device.
     *
     * @return <code>String</code> language locale, if specified
     */
    public String getAcceptLanguage () { return acceptLanguage; }

    /**
     * Retrieves the domain of the device's gateway.
     *
     * @return <code>String</code> domain of request origin (gateway domain)
     */
    public String getGateway() { return via; }

    /**
     * Retrieves a comma delimited list of file encodings supported by the device.
     *
     * @return <code>String</code> list of file encodings acceptable by the device.
     */
    public String getAcceptEncoding() { return acceptEncoding; }

    /**
     * Retrieves the list of Character Set encodings supported by the device.
     * <p>
     * The returned <code>String</code> may be contain more than one Character Set value,
     * in which case the the values will be returned as a comma delimeted list.
     *
     *  @return <code>String</code> character set(s) supported by this device.
     */
    public String getAcceptCharset() { return acceptCharset; }

    /**
```

```
     * Retrieves information describing device's network connection.
     * <p>
     * Typical format: {data transport type},{client IP address},{secure connection flag}
     *
     * @return <code>String</code> device's network information
     */
    public String getNetworkInfo() { return networkInfo; }

} // end
```

09847478.060501

```
/**
 * @(#)HTMLDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>      ↙
    factory for HTML devices
 */
public class HTMLDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -8489332563057202787L;


    /** name of this device profile */
    public static final String NAME = "TA_HTML";
    /** content type accepted by all devices matching this profile */
    protected static final String CONTENT_TYPE = "text/html";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile ↙
        creates.
     *
     * @return <code>Class</code> of the <code>HTMLDevice</code> instances that this profile ↙
        generates.
     */
    public Class getDeviceClass() {
        return new HTMLDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>   ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>HTMLDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>        ↙
        HTMLDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check User-Agent header
            HttpServletRequest request = (HttpServletRequest)req;
            String accept = request.getHeader("Accept");
            String userAgent = request.getHeader("User-Agent");

            return ( ((userAgent != null) && (userAgent.indexOf("Mozilla") >= 0) ||
                      (accept != null ) && (accept.indexOf("text/html") >= 0)) );
        }

        return false;
    }
    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        HTMLDevice device = new HTMLDevice (this);

        // no sure way of obtaining unique id, so do not set device.GUID

        // declare temp values for device properties, to check for null
        String  tempUserAgent,
                tempEncoding,
                tempHost,
                tempConnection,
                tempLanguage;

        // initialize temps
        tempUserAgent = request.getHeader("User-Agent");
        tempEncoding = request.getHeader ("Accept-Encoding");
        tempHost = request.getHeader("Host");
        tempConnection = request.getHeader("Connection");
        tempLanguage = request.getHeader("Accept-Language");


        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // ACCEPT-LANGUAGE
        device.acceptLanguage = (tempLanguage == null) ? STRING_NO_VALUE : tempLanguage;

        // HOST
        device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // ACCEPT-ENCODING
        device.acceptEncoding = (tempEncoding == null) ? STRING_NO_VALUE : tempEncoding;

        // CONNECTION TYPE
        device.connectionType = (tempConnection == null) ? STRING_NO_VALUE : tempConnection;


        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        HTMLDevice device = new HTMLDevice(this);

        // HTMLDevices have no GUID, so do not set

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.acceptLanguage = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.acceptEncoding = STRING_NO_VALUE;
        device.connectionType = STRING_NO_VALUE;

        return device;
    }

} // end
```

```
/**
 * @(#)HTMLDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for all HTML devices
 */
public class HTMLDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -937006866328419345L;

    protected String acceptLanguage, acceptEncoding, connectionType, host;

    /**
     * Constructs a new <code>HTMLDevice</code> instance of the type represented by profile.
     * @param profile <code>HTMLDeviceProfile</code> prototype for this device instance
     */
    HTMLDevice (HTMLDeviceProfile profile) {
        super(profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType() {
        return HTMLDeviceProfile.CONTENT_TYPE;
    }

    /**
     * Retrieves the language locale supported by the device.
     *
     * @return <code>String</code> language locale, if specified
     */
    public String getAcceptLanguage() { return acceptLanguage; }

    /**
     * Retrieves a comma delimited list of file encodings supported by the device.
     *
     * @return <code>String</code> list of file encodings acceptable by the device.
     */
    public String getAcceptEncoding() { return acceptEncoding; }

    /**
     * Retrieves the connection type for the request.
     *
     * @return <code>String</code> connection type for the request (typically "Keep-Alive" or
       "Close")
     */
    public String getConnectionType() { return connectionType; }

    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return host; }

} // end
```

```
/**
 * @(#)HDMLDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>
    factory for HDML devices
 */
public class HDMLDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -1427190061023906748L;


    /** name of this device profile */
    public static final String NAME = "TA_HDML";

    /** the content type expected by this device */
    static final String CONTENT_TYPE = "text/x-hdml";

    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile
        creates.
     *
     * @return <code>Class</code> of the <code>HDMLDevice</code> instances that this profile
        generates.
     */
    public Class getDeviceClass() {
        return new HDMLDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>HDMLDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>
        HDMLDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice(ServletRequest req) {

        if (super.isRequestFromDevice (req)) {
            // now cast request to HttpServletRequest to retrieve and check Accept header
            HttpServletRequest request = (HttpServletRequest)req;
            String accept = (String)request.getHeader ("Accept");

            return ( (accept != null ) && (accept.indexOf(CONTENT_TYPE) >= 0));
        }

        return false;
    }
    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
         requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest)req;
        HDMLDevice device = new HDMLDevice (this);

        // declare temp values for device properties, to check for null
        String  tempGUID,
                tempUserAgent;

        // initialize temps
        tempGUID = request.getHeader("x-up-subno");
        tempUserAgent = request.getHeader ("User-Agent");

        // **NEED TO VERIFY IF THIS IS ALWAYS UNIQUE** - gordogre 10/18/2000
        if (tempGUID != null) device.setGUID( HDMLDeviceProfile.NAME + ":" + tempGUID );

        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);


        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
         NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        HDMLDevice device = new HDMLDevice(this);

        device.setGUID( guid );

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);

        return device;
    }

} // end
```

```
/**
 * @(#)HDMLDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for all HDML devices
 */
public class HDMLDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 5976014305772814680L;

    /**
     * Constructs a new <code>HDMLDevice</code> instance of the type represented by profile.
     * @param profile <code>HDMLDeviceProfile</code> prototype for this device instance
     */
    HDMLDevice (HDMLDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType() {
        return HDMLDeviceProfile.CONTENT_TYPE;
    }

} // end
```

```
/**
 * @(#)GoWebRIMDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>  ↙
   factory
 * for GoWeb RIM devices
 */
public class GoWebRIMDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 6994915205364885 75L;

    /** name of this device profile */
    public static final String NAME = "TA_GOWEB_RIM";

    /** user-agent transmitted with every request coming from devices matching this profile  ↙
        */
    static final String USER_AGENT   = "Go.Web/1.1 (compatible; HandHTTP 1.1; Mozilla/1.0;  ↙
        RIM950; compatible )";

    /** content type accepted by all devices matching this profile */
    static final String CONTENT_TYPE = "text/vnd.wap.wml";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>GoWebRIMDevice</code> instances that this  ↙
        profile generates.
     */
    public Class getDeviceClass() {
        return new GoWebRIMDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>  ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }



    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>GoWebRIMDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>  ↙
        GoWebRIMDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice(req)) {
            // now cast request to HttpServletRequest to retrieve and test user-agent header
            HttpServletRequest request = (HttpServletRequest) req;
            String userAgent = (String) request.getHeader ("user-agent");
            return ((userAgent != null) && (userAgent.indexOf("Go.Web") >= 0) && (userAgent. ↙
                indexOf("RIM") >=0));
                // it's go web and it's rim, so return true...
        }

        return false;
    }


    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the ↙
        requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        GoWebRIMDevice device = new GoWebRIMDevice(this);

        // declare temp values for device properties, to check for null
        String  tempGUID,
                tempUserAgent,
                tempHost,
                tempReferer,
                tempLanguage;

        // initialize temps
        tempGUID = request.getHeader("x-ga-subno");
        tempUserAgent = request.getHeader ("User-Agent");
        tempHost = request.getHeader("Host");
        tempReferer = request.getHeader("Referer");
        tempLanguage = request.getHeader("Accept-Language");

        // **NEED TO VERIFY IF THIS IS ALWAYS UNIQUE** - gordogre 10/18/2000
        if (tempGUID != null) device.setGUID( GoWebRIMDeviceProfile.NAME + ":" + tempGUID );

        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // ACCEPT-LANGUAGE
        device.acceptLanguage = (tempLanguage == null) ? STRING_NO_VALUE : tempLanguage;

        // HOST
        device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // REFERER
        device.referer = (tempReferer == null) ? STRING_NO_VALUE : tempReferer;

        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate ↙
        NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        GoWebRIMDevice device = new GoWebRIMDevice(this);

        device.setGUID( guid );

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.acceptLanguage = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.referer = STRING_NO_VALUE;

        return device;
    }

} // end
```

```java
/**
 * @(#)GoWebRIMDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for all RIM devices with GoWeb browsers
 */
public class GoWebRIMDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 1155840689638109336L;

    protected String referer, host, acceptLanguage;

    /**
     * Constructs a new <code>GoWebRIMDevice</code> instance of the type represented by
       profile.
     * @param profile <code>GoWebRIMDeviceProfile</code> prototype for this device instance
     */
    GoWebRIMDevice (GoWebRIMDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType () {
        return GoWebRIMDeviceProfile.CONTENT_TYPE;
    }

    /**
     * Retrieves the domain of the device's gateway.
     *
     * @return <code>String</code> domain of request origin (gateway domain)
     */
    public String getReferer() { return referer; }

    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return host; }

    /**
     * Retrieves the language locale supported by the device.
     *
     * @return <code>String</code> language locale, if specified
     */
    public String getAcceptLanguage() { return acceptLanguage; }

} // end
```

```
/**
 * @(#)GoWebPalmDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>
   factory for GoWeb Palm devices
 */
public class GoWebPalmDeviceProfile extends HTTPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -5082728858823692866L;

    /** name of this device profile */
    public static final String NAME = "TA_GOWEB_PALM";

    /** content type accepted by all devices matching this profile */
    static final String CONTENT_TYPE = "text/vnd.wap.wml";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile
        creates.
     *
     * @return <code>Class</code> of the <code>GoWebPalmDevice</code> instances that this
        profile generates.
     */
    public Class getDeviceClass() { return new GoWebPalmDevice (this).getClass (); }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>GoWebPalmDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>
        GoWebPalmDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice (ServletRequest req) {

        if (super.isRequestFromDevice(req)) {
            // now cast request to HttpServletRequest to retrieve and test user-agent header
            HttpServletRequest request = (HttpServletRequest) req;
            String userAgent = (String) request.getHeader ("user-agent");

            return (userAgent != null && userAgent.indexOf("Go.Web") >= 0 && userAgent.
                indexOf("Palm") >=0);
                // it's go web and it's palm, so return true...
        }

        return false;
    }
```

```java
    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the ↙
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        GoWebPalmDevice device = new GoWebPalmDevice (this);

        // declare temp values for device properties, to check for null
        String  tempGUID,
                tempUserAgent,
                tempHost,
                tempReferer,
                tempLanguage;

        // initialize temps
        tempGUID = request.getHeader("x-ga-subno");
        tempUserAgent = request.getHeader ("User-Agent");
        tempHost = request.getHeader("Host");
        tempReferer = request.getHeader("Referer");
        tempLanguage = request.getHeader("Accept-Language");

        // **NEED TO VERIFY IF THIS IS ALWAYS UNIQUE** - gordogre 10/18/2000
        if (tempGUID != null) device.setGUID( GoWebPalmDeviceProfile.NAME + ":" + tempGUID );

        // COOKIES
        device.setCookies (request.getCookies ());

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent ( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // ACCEPT-LANGUAGE
        device.acceptLanguage = (tempLanguage == null) ? STRING_NO_VALUE : tempLanguage;

        // HOST
        device.host = (tempHost == null) ? STRING_NO_VALUE : tempHost;

        // REFERER
        device.referer = (tempReferer == null) ? STRING_NO_VALUE : tempReferer;

        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate ↙
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        GoWebPalmDevice device = new GoWebPalmDevice(this);
        device.setGUID( guid );

        // initialize properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.acceptLanguage = STRING_NO_VALUE;
        device.host = STRING_NO_VALUE;
        device.referer = STRING_NO_VALUE;

        return device;
    }

} // end
```

```
/**
 * @(#)GoWebPalmDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for all devices with GoWeb browsers on the Palm&#174; platform
 */
public class GoWebPalmDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 138821043664431198L;

    protected String referer, host, acceptLanguage;

    /**
     * Constructs a new <code>GoWebPalmDevice</code> instance of the type represented by
       profile.
     * @param profile <code>GoWebPalmDeviceProfile</code> prototype for this device instance
     */
    GoWebPalmDevice (GoWebPalmDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device.
     */
    public String getContentType() {
        return GoWebPalmDeviceProfile.CONTENT_TYPE;
    }

    /**
     * Retrieves the domain of the device's gateway.
     *
     * @return <code>String</code> domain of request origin (gateway domain)
     */
    public String getReferer() { return referer; }

    /**
     * Retrieves the address of the host targeted by the request.
     *
     * @return <code>String</code> host targeted by request
     */
    public String getHost() { return host; }

    /**
     * Retrieves the language locale supported by the device.
     *
     * @return <code>String</code> language locale, if specified
     */
    public String getAcceptLanguage() { return acceptLanguage; }

} // end
```

```
/**
 * @(#)EricssonWAPDeviceProfile.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>      ↙
   factory for Ericsson WAP phones.
 */


public class EricssonWAPDeviceProfile extends WAPDeviceProfile {


    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -9152178982246895559L;

    /** name of this device profile */
    public static final String NAME = "TA_ERICSSON_WAP";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>EricssonWAPDevice</code> instances that this  ↙
        profile generates.
     */
    public Class getDeviceClass() {
        return new EricssonWAPDevice(this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>   ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>EricssonWAPDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>         ↙
        EricssonWAPDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice(ServletRequest req) {

        if (super.isRequestFromDevice(req)) {
            // now cast request to HttpServletRequest to retrieve and test user-agent header
            HttpServletRequest request = (HttpServletRequest) req;
            String userAgent = (String) request.getHeader ("user-agent");

            return (userAgent != null &&
                    ( (userAgent.indexOf("Ericsson") >= 0) ||
                      (userAgent.indexOf("R380") >= 0) ||
                      (userAgent.indexOf("R320") >= 0) ||
                      (userAgent.indexOf("888") >= 0) ) );
        }

        return false;
    }


    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the
       requesting device.
     */
    public Device createDeviceFromRequest (ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        EricssonWAPDevice device = new EricssonWAPDevice(this);

        // FIXME FIXME FIXME
        // sgross 6/14/2000: is this unique?
        // this is at best the device's IP address, not reliably unique - gordogre 10/18/2000
        // also need to check for null to avoid null pointer exception on substring(5)
        String tempGUID = request.getHeader ("x-network-info");
        if (tempGUID != null) device.setGUID( NAME + ":"+ tempGUID.substring(5) );


        // declare temp values for device properties, to check for null
        String  tempUserAgent,
                tempCharset,
                tempGateway,
                tempLanguage;

        // initialize temps
        tempUserAgent = request.getHeader ("user-agent");
        tempCharset = request.getHeader("accept-charset");
        tempGateway = request.getHeader("via");
        tempLanguage = request.getHeader("accept-language");

        // COOKIES
        device.setCookies (request.getCookies ());
        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // USER-AGENT
        device.setUserAgent( (tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent);

        // CHAR-SET
        device.charset = (tempCharset == null) ? STRING_NO_VALUE : tempCharset;

        // GATEWAY
        device.gateway = (tempGateway == null) ? STRING_NO_VALUE : tempGateway;

        // ACCEPT LANGUAGE
        device.language = (tempLanguage == null) ? STRING_NO_VALUE : tempLanguage;

        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {

        EricssonWAPDevice device = new EricssonWAPDevice(this);

        device.setGUID( guid );

        // initialize properties to NO_VALUE
        device.setUserAgent(STRING_NO_VALUE);
        device.charset = STRING_NO_VALUE;
        device.gateway = STRING_NO_VALUE;
        device.language = STRING_NO_VALUE;

        return device;
    }

} // end
```

```
/**
 * @(#)EricssonWAPDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 *  ThinAirApps' abstraction for the Ericcson family of WAP phones.
 */
public class EricssonWAPDevice extends WAPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -7071206479514820282L;

    protected String host, charset, gateway, language;

    /**
     * Constructs a new <code>EricssonWapDevice</code> instance from this profile.
     *
     * @param profile <code>EricssonWapDeviceProfile</code> prototype used to create device.
     */
    EricssonWAPDevice(EricssonWAPDeviceProfile profile) {
        super(profile);
    }

    /**
     * Retrieves the language locale supported by the device.
     *
     *  @return <code>String</code> language locale supported by client device.
     */
    public String getLanguage() { return language; }

    /**
     * Retrieves the list of Character Set encodings supported by the device.
     * <p>
     * The returned <code>String</code> may be contain more than one Character Set value,
     * in which case the the values will be returned as a comma delimeted list.
     *
     *  @return <code>String</code> character set(s) supported by this device.
     */
    public String getCharset() { return charset; }

    /**
     * Retrieves a description of the WAP gateway used by this device.
     *
     *  @return <code>String</code> version of gateway used by this device
     */
    public String getGateway() { return gateway; }

} // end
```

```
/**
 * @(#)AvantGoDeviceProfile.java
 */
package com.thinairapps.platform.device;

import com.thinair.utils.*;

import javax.servlet.ServletRequest;
import javax.servlet.http.HttpServletRequest;


/**
 * implements ThinAirApps's standard <code>DeviceProfile</code> and <code>Device</code>      ↙
    factory for AvantGo devices
 */
public class AvantGoDeviceProfile extends HTTPDeviceProfile {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = 3548305456343669735L;

    /** name of this device profile */
    public static final String NAME = "TA_AVANTGO";

    /** user-agent transmitted with every request coming from devices matching this profile  ↙
        */
    static final String USER_AGENT   = "Mozilla/3.0 (compatible; AvantGo 3.2)";

    /** content type accepted by all devices matching this profile */
    static final String CONTENT_TYPE = "text/html";


    /**
     * Retrieves the <code>Class</code> of the <code>Device</code> object that this profile  ↙
        creates.
     *
     * @return <code>Class</code> of the <code>AvantGoDevice</code> instances that this      ↙
        profile generates.
     */
    public Class getDeviceClass() {
        return new AvantGoDevice (this).getClass ();
    }


    /**
     * Retrieves the friendly name of this device type description.
     *
     * @return <code>String</code> A friendly name callers can use to refer to this <code>   ↙
        DeviceProfile</code>
     */
    public String getName() { return NAME; }


    /**
     * Determines whether the actual device making a request is of the type represented by
     * <code>AvantGoDeviceProfile</code>.
     *
     * @param request the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>boolean</code> true if the  requesting device is of type <code>         ↙
        AvantGoDevice</code>, false otherwise.
     */
    public boolean isRequestFromDevice(ServletRequest req) {

        // check if device is an HTTP device
        if (super.isRequestFromDevice(req)) {
            // cast req to HttpServletRequest to retrieve and check User-Agent
            HttpServletRequest request = (HttpServletRequest) req;
            String userAgent = (String) request.getHeader("User-Agent");
            // cannot check for equivalence with USER_AGENT constant since other versions of ↙
                AvantGo are possible - gordogre
            return (userAgent != null && userAgent.indexOf("AvantGo") >=0);
        } else
            return false;
    }


    /**
     * Create a <code>Device</code> instance with the same properties as the actual
     * physical device that generated this servlet request.
     *
     * @param req the actual <code>ServletRequest</code> received by the ThinAir Server
     *
     * @return <code>Device</code> a device instance with properties set to describe the ↙
         requesting device.
     */
    public Device createDeviceFromRequest(ServletRequest req) {

        HttpServletRequest request = (HttpServletRequest) req;
        AvantGoDevice device = new AvantGoDevice (this);

        // declare temps for device properties for null checking
        String  tempDeviceId,
                tempColorDepth,
                tempScreenSize,
                tempDeviceOS,
                tempUserID,
                tempUserAgent,
                tempVersion,
                tempClientIP;

        // initialize temp properties
        tempDeviceId = request.getHeader ("X-AvantGo-DeviceId");
        tempColorDepth = request.getHeader ("X-AvantGo-ColorDepth");
        tempScreenSize = request.getHeader ("X-AvantGo-ScreenSize");
        tempDeviceOS = request.getHeader ("X-AvantGo-DeviceOS");
        tempUserID = request.getHeader ("X-AvantGo-UserId");
        tempUserAgent = request.getHeader ("User-Agent");
        tempVersion = request.getHeader("X-AvantGo-Verson");
        tempClientIP = request.getHeader("Client-ip");


        // the following 5 headers are base64 encoded, so we must first check for their ↙
            presence before
        // decoding to avoid any null pointer exceptions

        // DeviceId  may no longer be valid (10/17 gordogre)
        if (tempDeviceId != null) device.setGUID( NAME + ":" + Base64.decode(tempDeviceId));

        // ACCEPT
        device.setAccept(request.getHeader("Accept"));

        // COOKIES
        device.setCookies (request.getCookies());

        // USER AGENT
        device.setUserAgent ((tempUserAgent == null) ? STRING_NO_VALUE : tempUserAgent );

        // COLOR DEPTH
        device.colorDepth = (tempColorDepth == null) ? STRING_NO_VALUE :  Base64.decode ↙
            (tempColorDepth);

        // SCREEN SIZE
        device.screenSize = (tempScreenSize == null) ? STRING_NO_VALUE : Base64.decode ↙
            (tempScreenSize);

        // DEVICE OS
        device.deviceOS = (tempDeviceOS == null) ? STRING_NO_VALUE : Base64.decode ↙
```

```
            (tempDeviceOS);

        // USER ID
        device.userID =(tempUserID == null) ? STRING_NO_VALUE : Base64.decode(tempUserID);

        // VERSION
        // if AvantGo version 3.1 is being used, X-AvantGo-Version will not be present -     ↙
            gordogre
        device.version = (tempVersion == null) ? STRING_NO_VALUE : tempVersion;

        // CLIENT IP
        // clientIP may no longer be valid - gordogre
        device.clientIP = (tempClientIP == null) ? STRING_NO_VALUE : tempClientIP;

        // SecureSync and onlineRequest headers only present when true,
        // so a value is always set (i.e. there is no "NO_VALUE" for these properties)
        device.secureSync = (request.getHeader("X-AvantGo-SecureSync") != null) ? true :     ↙
            false;
        device.onlineRequest = (request.getHeader("X-AvantGo-OnlineRequest") != null) ? true ↙
            : false;

        return device;
    }


    /**
     * Create a <code>Device</code> from a <code>String</code> device GUID
     * This method is used primarily by administrators to preconfigure an account
     * to include a device.
     *<p>
     * This method initializes all device properties other than GUID to the appropriate    ↙
       NO_VALUE constant.
     *
     * @param guid unique device ID - may be null
     *
     * @return a <code>Device</code> object representing an actual device
     */
    public Device createDevice(String guid) {
        AvantGoDevice device = new AvantGoDevice(this);
        device.setGUID( guid );

        // initialize all properties to NO_VALUE
        device.setUserAgent (STRING_NO_VALUE);
        device.colorDepth = STRING_NO_VALUE;
        device.screenSize = STRING_NO_VALUE;
        device.deviceOS = STRING_NO_VALUE;
        device.userID = STRING_NO_VALUE;
        device.version = STRING_NO_VALUE;
        device.clientIP = STRING_NO_VALUE;

        return device;
    }


} // end
```

```
/**
 * @(#)AvantGoDevice.java
 */
package com.thinairapps.platform.device;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * ThinAirApps's abstraction for all devices with AvantGo browser version 3.2
 */
public class AvantGoDevice extends HTTPDevice {

    // serialVersionUID for compatability with previous versions
    static final long serialVersionUID = -2822887137691798848L;

    protected String version, colorDepth, screenSize, deviceOS, userID, clientIP;
    boolean onlineRequest,secureSync;

    /**
     * Constructs a new <code>AvantGoDevice</code> instance of the type represented by
        profile.
     * @param profile <code>AvantGoDeviceProfile</code> prototype for this device instance
     */
    AvantGoDevice (AvantGoDeviceProfile profile) {
        super (profile);
    }

    /**
     * Retrieves the content encoding type supported by this device.
     *
     * @return <code>String</code> content type supported by this device
     * */
    public String getContentType() { return AvantGoDeviceProfile.CONTENT_TYPE; }

    /**
     * Retrieves the version of the AvantGo browser this device is running.
     *
     * @return <code>String</code> client verison number
     */
    public String getVersion() { return version; }

    /**
     * Retrieves the color depth of the screen for the device.
     *
     * @return <code>String</code> color depth of device
     */
    public String getColorDepth() { return colorDepth; }

    /**
     * Retrieves the screen size of the device.
     * <p>
     * The returned value will be in the format "HxW", where H is the height of the screen
     * in pixels, and W is the width of the screen in pixels.
     *
     * @return <code>String</code> screen size of the device.
     */
    public String getScreenSize() { return screenSize; }

    /**
     * Retrieves a description of the operating system running on the device.
     *
     *  @return <code>String</code> get device operating system
     */
    public String getOS() { return deviceOS; }

    /**
     * Retrieves the AvantGo User Id for the user making the request.
     *  @return <code>String</code> AvantGo user id for person using this device
     */
    public String getUserId() { return userID; }

    /**
     * Retrieves logical flag indicating an online request.
     * <p>
     * Devices using the AvantGo browser may be making an online request or an offline   ↙
       request.
     * When a request is an offline request, the AvantGo application caches the contents of ↙
       the requested
     * page for offline viewing on a device following a device synchronization. When a    ↙
       request is an online
     * request, the user is interacting with the requested material in real time.
     *
     *  @return <code>boolean</code> true if request is an online request, false if request ↙
       is an offline request.
     */
    public boolean isOnlineRequest() { return onlineRequest; }

    /**
     * Retrieves logical flag indicating that a request has been made via a secure connection↙
       .
     *
     *  @return <code>boolean</code> true if request is using a secure connection, false    ↙
       otherwise.
     */
    public boolean isSecureSync() { return secureSync; }

    /**
     * Retrieves the IP address of the device.
     * <p>
     * This value may not be available for all requests, and may not be unique for all    ↙
       devices.
     *
     * @return <code>String</code> ip address of client device */
    public String getClientIP() { return clientIP; }

} // end
```

Getting Started
the Hello World Sample Connector

Wireless SDK for ThinAir Server

About this Sample

This is a very simple 'Hello World' Connector that serves as an introduction to building custom applications using the the ThinAir Connector API. The Hello World Connector recognizes what device is contacting it and returns a simple message in the markup language for that device. This Connector demonstrates:

1. A simple use of the ThinAirServer's Device detection ability
2. The use of different Tag libraries to render markup specific to each device
3. The ability for a Connector to process configuration file information. The Connector will append the value for SecretMessage in connector.ini to its output.

Requirements

This sample requires the following SDK JARs:

* platform.jar
* taglib.jar
* devices.jar

Sample Files

This sample consists of the following file tree:

connector.ini - connector configuration file

HelloWorldConnector.class - compiled Java code

/src - Java source files

Building the Sample

Compile the sample code using the Java compiler of your choice. Make sure to append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a subdirectory of the ThinAir Server's /Connectors subdirectory, given a name of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.

Using the Sample

Wait until the ThinAirServer has started and the Hello World Connector has been loaded and initialized. From your wireless device, enter the IP address listed as the value for ApplicationPath in connector.ini (your ThinAirServer IP address), followed by /samples/helloworld. For a machine with IP address 111.222.12.34 this would be:

```
    http://111.222.12.34/samples/helloworld
Follow the on-screen instructions.

===============================================================================
                        Last updated: 11.13.2000
                  Copyright 1999, 2000 ThinAirApps Inc.
===============================================================================
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//ThinAir Platform imports
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;

//ThinAir Tag Libraries imports
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;
import com.thinairapps.tag.html.*;
import com.thinairapps.tag.hdml.*;

//Standard Java imports
import java.util.*;
import java.io.*;

/*
 * This is a sample Connector Application for use with ThinAir Server
 * It's purpose is to demonstrate how to write a simple connector, that
 * is able to use the device determination capabilites of the platform.
 *
 */
public class HelloWorldConnector implements Connector
{

    //private members for the application's user
    private String SecretMessage = null;

   /**
    * init() is called by the ThinAirServer when the Connector is loaded.  It provides
    * the Connector with resources it needs to interact with the ThinAirServer.
    *
    * @param applicationName friendly name of the application
    * @param applicationPath URL path on which the server is hosting this Connector
    * @param appProps properties loaded from the "connector.ini" file
    * @param ca ConnectorAccess object to support profiles, sessions, and provider access
    * @param applicationLog is used for Logging
    */
    public void init (String applicationName, String applicationPath, Properties appProps,    ↙
        ConnectorAccess ca, ApplicationLog al)
    {
        //load "SecretMessage" property from "connector.ini" file
        SecretMessage = appProps.getProperty("SecretMessage");
    }


    /**The handle method implements the core logic of a Connector.  It takes an incoming    ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,    ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props Device's request properties (i.e. http GET or POST arguments)
     * @param device Device making the request
     * @param out OutputStream to write results to
     */
    public void handle (Properties props, Device device, OutputStream out) throws IOException
    {
        // hold the return markup
        String result = null;

        // generate a simple message in the rendering format of the requesting device

        if (device instanceof WAPDevice)
            {
                // WML parsing WAP device
                WMLTagDocument deck = new WMLTagDocument();
                com.thinairapps.tag.wml.Head head = new com.thinairapps.tag.wml.Head();
                com.thinairapps.tag.wml.Meta meta = new com.thinairapps.tag.wml.Meta          ↙
                    ("http-equiv", "Cache-Control", "max-age=0");
                head.addChild(meta);
                deck.addChild(head);

                com.thinairapps.tag.wml.DisplayCard card = new com.thinairapps.tag.wml.       ↙
                    DisplayCard("HelloWorld", "Hello World");
                card.buildCard("Hello WAP User! "+SecretMessage, com.thinairapps.tag.wml.     ↙
                    Paragraph.ALIGN_LEFT);
                deck.addChild(card);

                result = deck.render();
            }
            else if (device instanceof HDMLDevice)
            {
                // HDML parsing HTTP device
                HDMLTagDocument deck = new HDMLTagDocument();
                com.thinairapps.tag.hdml.DisplayCard card = new com.thinairapps.tag.hdml.     ↙
                    DisplayCard();
                card.addChild(new FormattedLine("Hello HDML User! "+SecretMessage,            ↙
                    FormattedLine.LEFT));
                deck.addChild(card);

                result = deck.render();
            }
                else
                {
                    // HTML parsing HTTP device
                    HTMLTagDocument doc = new HTMLTagDocument();
                    com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
                    com.thinairapps.tag.html.Meta meta = new com.thinairapps.tag.html.Meta    ↙
                        ("name", "PalmComputingPlatform", "true");
                    head.addChild(meta);
                    com.thinairapps.tag.html.Title title = new com.thinairapps.tag.html.Title↙
                        ("Hello World");
                    head.addChild(title);
                    doc.setHead(head);

                    com.thinairapps.tag.html.Body body = new com.thinairapps.tag.html.Body();
                    com.thinairapps.tag.html.Font font = new com.thinairapps.tag.html.Font    ↙
                        ("geneva,arial", 3);

                    com.thinairapps.tag.html.Text text;
                    if (device instanceof GoWebRIMDevice)
                        text = new com.thinairapps.tag.html.Text("Hello GoWeb User! "+        ↙
                            SecretMessage);
                    else
                        text = new com.thinairapps.tag.html.Text("Hello HTTP User! "+         ↙
                            SecretMessage);

                    font.addChild( text );
                    body.addChild(font);
                    doc.addChild(body);

                    result = doc.render();
            }

        // write result to the output stream
        out.write(result.getBytes());
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a        ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing   ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used   ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample    ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this        ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String devices[] =
        {
            UPWAPDeviceProfile.NAME, PalmVIIDeviceProfile.NAME,
            AvantGoDeviceProfile.NAME, HDMLDeviceProfile.NAME,
            GoWebRIMDeviceProfile.NAME, HTMLDeviceProfile.NAME
        };

        return devices;
    }
}
```

DeviceDetective
a.k.a. Inspector Gadget
Sample Connector
Wireless SDK for ThinAir Server

About this Sample

The DeviceDetective sample Connector:

1. Demonstrates how to use the Device detection features of the ThinAir Server

2. Showcases the Device objects and how they function in a working Connector

3. Demonstrates the various Tag Libraries that render markup for different devices using a simple, uniform API

With this Connector you will be able to contact your ThinAirServer with almost any wireless device, and have it automatically map your actual device to one of several built-in 'ThinAir Device Profiles'. The Device Profiles will generate a Device object with the same properties as your actual device. Properties might include things like screen size, color depth, and number of soft keys. The Connector will return a page, in your device's own markup language, listing the properties it detected along with the Device's name.

For instructions on how to build your own Device Profiles, consult the Developer's Guide.

Requirements

This sample requires the following SDK JARs:

* platform.jar
* taglib.jar
* devices.jar

Sample Files

This sample consists of the following file tree:

connector.ini - connector configuration file

DeviceDetective.jar - compiled Java code

/src - Java source files

Building the Sample

Compile the sample code using the Java compiler of your choice. Make sure to append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a subdirectory of the ThinAir Server's /Connectors subdirectory, given a name of your choice.

Start the ThinAir Server; it will load the sample code and begin executing it.

Using the Sample

Wait until the ThinAirServer has started and the Device Detective Connector has been loaded and initialized. From your wireless device, enter the IP address listed as the value for ApplicationPath in connector.ini (your ThinAirServer IP address), followed by /samples/device. For a machine with IP address 111.222.12.34 this would be:

```
     http://111.222.12.34/samples/device
```

Supported devices include: WAP phones, HDML phones, Palm Pilots, Windows CE devices, desktop web browsers, and GO America/GO RIM pagers. In order to build a Palm Query Application (PQA) that works with DeviceDetective, you will need to understand and use "Web Clipping" technology from Palm. Web Clipping involves essentially creating HTML interfaces into your applications. For your convienence, an HTML file (deviceDetective.html) has been provided for this purpose. To find out more about creating PQAs and Web Clipping technology, visit: http://www.palmos.com/dev/tech/webclipping/

==

Last updated: 11.13.2000

==

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//ThinAir device library imports
import com.thinairapps.platform.device.*;

//ThinAir Tag Libraries imports
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

//Standard Java imports
import java.util.*;



/**
 * This utility class renders output as WML for a variety of devices
 */

class WMLRenderer
{

    /**
     * Build a card with a welcome message that identifies the device to the user
     *
     * @param deviceName String of Device name to be displayed
     * @param keys[] String array of keys to be displayed
     * @param values[] String array of values to be displayed
     *
     * @return String of formatted WML
     */
    static String buildCard(String deviceName, String keys[], String values[])
    {

        WMLTagDocument deck = new WMLTagDocument();

        Head head = new Head();
        Meta meta = new Meta("http-equiv", "Cache-Control", "max-age=0");

        // this is to test cache control at the deck level...
        double rnd = Math.random();
        Meta meta2 = new Meta("name", String.valueOf(rnd), "max-age=0");
        head.addChild(meta);
        head.addChild(meta2);
        deck.addChild(head);

        DisplayCard card = new DisplayCard("device", "Inspector Gadget");
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER, Paragraph.MODE_NOWRAP);

        StringBuffer sb = new StringBuffer(56);
        sb.append("<b>Welcome ");
        sb.append(deviceName);
        sb.append("</b>");


        p.addChild( new Text(sb.toString().trim()) );
        card.addChild(p);

        // print out the properties of the device
        // as key: value pairs
        p = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);
        p.addChild(new Text("Device Properties:<br/>"));

        Break br = new Break();
        for (int i = 0; i < keys.length; i++)
        {
            p.addChild(new Text(keys[i] + ": " + values[i]));
            p.addChild(br);
        }

        card.addChild(p);

        deck.addChild(card);
        return deck.render();
    }



    /**
     * @param device used to retrieve WAPdevice values
     *
     * @return page welcoming a user of a basic WAP device
     */
    static String getMessage(WAPDevice device)
    {

        String keys[] = new String[4];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";

        String values[] = new String[4];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();

        return buildCard( device.getProfile().getName(), keys, values);
    }



    /**
     * @param device used to retrieve GoWebRimDevice values
     *
     * @return page welcoming a user of a GoWeb device
     */
    static String getMessage(GoWebRIMDevice device)
    {

        String keys[] = new String[4];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";

        String values[] = new String[4];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();

        return buildCard( device.getProfile().getName(), keys, values);
    }



    /**
     * @parm device used to retrieve Unwired Planet WAP device values
     *
     * @return page welcoming a user of a basic UP/WAP device
     */
    static String getMessage(UPWAPDevice device)
    {
        String keys[] = new String[14];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";
        keys[4] = "Language";
        keys[5] = "Smart Dialing";
        keys[6] = "Screen Depth";
        keys[7] = "Color";
        keys[8] = "Alert";
        keys[9] = "Max PDU";
        keys[10] = "Soft Keys";
        keys[11] = "Screen Chars";
        keys[12] = "Pixel Width";
        keys[13] = "Pixel Height";


        String values[] = new String[14];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();
        values[4] = device.getLanguage();
        values[5] = String.valueOf(device.isSmartDialing());
        values[6] = String.valueOf(device.getScreenDepth());
        values[7] = String.valueOf(device.isColor());
        values[8] = String.valueOf(device.immediateAlert());
        values[9] = device.getMaxPDU();
        values[10] = String.valueOf(device.numSoftKeys());
        values[11] = device.getScreenChars();
        values[12] = String.valueOf(device.getPixelWidth());
        values[13] = String.valueOf(device.getPixelHeight());

        return buildCard( device.getProfile().getName(), keys, values);
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//ThinAir device library imports
import com.thinairapps.platform.device.*;

//ThinAir Tag Libraries imports
import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;

//Standard Java imports
import java.util.*;

/*
 * This utility class renders output as HTML for a variety of devices
 */
class HTMLRenderer {

    /**
     * Build a card with a welcome message that identifies the device to the user
     *
     * @param deviceName String of Device name to be displayed
     * @param keys[] String array of keys to be displayed
     * @param values[] String array of values to be displayed
     *
     * @return String of formatted HTML
     */
    static String buildCard(String deviceName, String keys[], String values[])
    {

        HTMLTagDocument doc = new HTMLTagDocument();

        Head head = new Head();
        Meta meta = new Meta("name", "PalmComputingPlatform", "true");
        head.addChild(meta);
        head.addChild(new Title("Inspector Gadget"));

        doc.setHead(head);

        Body body = new Body();
        Font font = new Font("geneva,arial", 3);

        Center center = new Center();

        StringBuffer sb = new StringBuffer(56);
        sb.append("<B>Welcome ");
        sb.append(deviceName);
        sb.append("</B>");
        center.addChild( new Text(sb.toString().trim()) );

        font.addChild(center);
        font.addChild(new Break());

        // print out the properties of the device
        // as key: value pairs
        font.addChild(new Text("Device Properties:<BR>"));
        Break br = new Break();
        for (int i = 0; i < keys.length; i++)
        {
            font.addChild(new Text(keys[i] + ": " + values[i]));
            font.addChild(br);
        }

        body.addChild(font);
        doc.setBody(body);
        return doc.render();
    }



    /**
     *
     * @parm device used to retrieve HTML device values
     *
     * @return String consisting of a page welcoming a user of a basic HTML device
     */
    static String getMessage(HTMLDevice device)
    {

        String keys[] = new String[4];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";

        String values[] = new String[4];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();

        return buildCard( device.getProfile().getName(), keys, values);
    }



    /**
     *
     * @param device used to retrieve values from PalmVIIDevice
     *
     * @return page welcoming a user of a PalmVIIDevice device
     */
    static String getMessage(PalmVIIDevice device)
    {

        String keys[] = new String[4];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";

        String values[] = new String[4];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();

        return buildCard( device.getProfile().getName(), keys, values);
    }



    /**
     *
     * @param device used to retrieve values from AvantGoDevice
     *
     * @return page welcoming user of AvantGoDevice
     */
    static String getMessage(AvantGoDevice device)
    {
        String keys[] = new String[11];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";
        keys[4] = "Version";
        keys[5] = "Color Depth";
        keys[6] = "Screen Size";
        keys[7] = "OS";
        keys[8] = "UserID";
        keys[9] = "Online";
        keys[10] = "IP";

        String values[] = new String[11];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();
        values[4] = device.getVersion();
        values[5] = device.getColorDepth();
        values[6] = device.getScreenSize();
        values[7] = device.getOS();
        values[8] = device.getUserId();
        values[9] = String.valueOf( device.isOnlineRequest() );
        values[10] = device.getClientIP();

        return buildCard( device.getProfile().getName(), keys, values);
    }


    /**
     * @param device used to retrieve values from PocketIEDevice
     *
     * @return page welcoming user of PocketIEDevice
     */
    static String getMessage(PocketIEDevice device)
    {
        String keys[] = new String[7];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";
        keys[4] = "Color Depth";
        keys[5] = "Screen Size";
        keys[6] = "OS";

        String values[] = new String[7];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();
        values[4] = device.getColorDepth();
        values[5] = device.getScreenSize();
        values[6] = device.getOS();

        return buildCard( device.getProfile().getName(), keys, values);
    }
}
```

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//ThinAir device library imports
import com.thinairapps.platform.device.*;

//Thinair tag libraries imports
import com.thinairapps.tag.*;
import com.thinairapps.tag.hdml.*;

//Standard Java imports
import java.util.*;

/**
 * This utility class renders output as HDML for a variety of devices
 */
class HDMLRenderer
{

    /**
     * Build a card with a welcome message that identifies the device to the user
     *
     * @param deviceName String of Device name to be displayed
     * @param keys[] String array of keys to be displayed
     * @param values[] String array of values to be displayed
     *
     * @return String of formatted HDML
     */
    static String buildCard(String deviceName, String keys[], String values[])
    {

        HDMLTagDocument deck = new HDMLTagDocument();

        DisplayCard card = new DisplayCard();

        StringBuffer sb = new StringBuffer(56);
        sb.append("Welcome ");
        sb.append(deviceName);
        card.addText(new FormattedLine( sb.toString().trim(), FormattedLine.CENTER));
        card.addChild(new Break());


        card.addChild(new FormattedLine("Device Properties:<BR>", FormattedLine.LEFT));
        Break br = new Break();
        for (int i = 0; i < keys.length; i++)
        {
            card.addChild(new FormattedLine(keys[i] + ": " + values[i], FormattedLine.LEFT));
            card.addChild(br);
        }

        deck.addCard(card);
        return deck.render();
    }



    /**
     * Retrive values from HDML device and generates page
     *
     * @parm device used to retrieve HDML device values
     *
     * @return String consisting of a page welcoming a user of a basic HDML device
     */
    static String getMessage(HDMLDevice device)
    {
        String keys[] = new String[4];
        keys[0] = "GUID";
        keys[1] = "Protocol";
        keys[2] = "Content-Type";
        keys[3] = "User-Agent";

        String values[] = new String[4];
        values[0] = device.getGUID();
        values[1] = device.getProtocol();
        values[2] = device.getContentType();
        values[3] = device.getUserAgent();

        return buildCard( device.getProfile().getName(), keys, values);
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Thinairapps Imports
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;

//Standard Java Imports
import java.net.*;
import java.io.*;
import java.util.*;


/**
 * This Connector demonstrates ThinAir Server's facilities for detecting devices and   ↙
    rendering
 * content conditionally using different Tag Libraries
 */
public class DeviceDetective implements Connector
{

    private ConnectorAccess connectorAccess;
    private String path;


    /**Called by the ThinAirServer when the Connector is loaded.  It provides the Connector  ↙
        with
    * resources it needs to interact with the ThinAirServer.
    *
    * For more information about the Connector interface, see the javadocs for the ThinAir  ↙
        Server API
    *
    * @param applicationName is a String derived from connector.ini.
    * @param applicationPath is a String derived from connector.ini.
    * @param connectorProps is a Properties list containing developer assigned        ↙
        connector-specific properties.
    * @param connectorAccess is our access point to the services provided by ThinAir Server.
    * @param applicationLog is used for logging
    */
    public void init(String name, String p, Properties iniProps, ConnectorAccess ca,   ↙
        ApplicationLog al)
    {
        connectorAccess = ca;
            path = path;
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a   ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing  ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used  ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample  ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this     ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String devices[] =
        {
            AvantGoDeviceProfile.NAME,
            GoWebPalmDeviceProfile.NAME,
            GoWebRIMDeviceProfile.NAME,
            HDMLDeviceProfile.NAME,
            HTMLDeviceProfile.NAME,
            NokiaWAPDeviceProfile.NAME,
            PalmVIIDeviceProfile.NAME,
            PocketIEDeviceProfile.NAME,
            UPWAPDeviceProfile.NAME,
            WAPDeviceProfile.NAME
        };
        return devices;
    }


    /**The handle method implements the core logic of a Connector.  It takes an incoming ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,  ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters  ↙
        from the device.
     * @param device a Device object created in the image of the actual device making this  ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties reqProps, Device device, OutputStream out) throws        ↙
        IOException
    {
        // hold the return markup
        String result = null;

        try
        {
            // determine which device is contacting the Connector and use a different
            // rendering class to generate output
            if (device instanceof HDMLDevice)
                result = HDMLRenderer.getMessage( (HDMLDevice) device );

            else if (device instanceof AvantGoDevice)
                result = HTMLRenderer.getMessage( (AvantGoDevice) device );

            else if (device instanceof GoWebPalmDevice)
                result = HTMLRenderer.getMessage( (HTMLDevice) device );

            else if (device instanceof PocketIEDevice)
                result = HTMLRenderer.getMessage( (PocketIEDevice) device );

            else if (device instanceof PalmVIIDevice)
                result = HTMLRenderer.getMessage( (PalmVIIDevice) device );

            else if (device instanceof HTMLDevice)
                result = HTMLRenderer.getMessage( (HTMLDevice) device );
            else if (device instanceof GoWebRIMDevice)
                result = WMLRenderer.getMessage( (GoWebRIMDevice) device );

            else if (device instanceof NokiaWAPDevice)
                result = WMLRenderer.getMessage( (NokiaWAPDevice) device );

            else if (device instanceof UPWAPDevice)
                result = WMLRenderer.getMessage( (UPWAPDevice) device );

            else if (device instanceof WAPDevice)
                result = WMLRenderer.getMessage( (WAPDevice) device );

            else
                result = "ERROR: Device "+device.getClass()+" not supported.";

        }
        catch (Exception e)
        {
            e.printStackTrace();
            result = "ERROR: "+e.getMessage();
        }

        out.write( result.getBytes() );
    }
}
```

Inspector Gadget

Press GO and I will discover your device automatically:

GO

09874378.060601

```
==============================================================================

                      Database Connector Sample Connector

                      Wireless SDK for ThinAir Server

==============================================================================


-----------------
About this Sample
-----------------

This sample connector demonstrates the ability to connect to a Database using a
ODBC connection, retrieve data and display it on either a HTML or WML device.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * device.jar


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    DBConnector.class - compiled Java code

    /src - java source file - DBconnector.java


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  Make sure to
append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Create a ODBC DSN name of "Northwind" pointing to the Microsoft Access
Northwind Database.


Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the DBconnector has
been loaded and initialized.  From your wireless WML device, or web browser,
enter the IP address listed as the value for ApplicationPath in connector.ini
(your ThinAirServer IP address), followed by /samples/DBconnector.  For a
machine with IP address 111.222.12.34 this would be:

     http://111.222.12.34/samples/DBconnector

Follow the on-screen instructions.

==============================================================================

                          Last updated: 11.13.2000
```

```
===============================================================================
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//ThinAir Platform import
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;

//ThinAir Tag Libraries import
import com.thinairapps.tag.html.*;
import com.thinairapps.tag.wml.*;

import java.io.*;
import java.util.*;
import java.sql.*;



/*
 * This is a sample Connector Application for use with ThinAir Server 1.2. Its
 * purpose is to demonstrate a simple connection to a database utilizing the
 * ThinAir Connector API. A ODBC connection to the Microsoft Access Northwind
 * Database needs to be established with the DSN name of "Northwind".
 */

public class DBconnector implements Connector
{

    private String appname;
    private ConnectorAccess connectaccess;


     /**
      * init() is called by the ThinAirServer when the Connector is loaded.  It
      * provides the Connector with resources it needs to interact with the
      * ThinAir Server. For more information about the Connector interface, see
      * the javadocs for the ThinAir Server API
      *
      * @param applicationName is a String derived from connector.ini.
      * @param applicationPath is a String derived from connector.ini.
      * @param connectorProps is a Properties list containing developer assigned
      *                      connector-specific properties.
      * @param connectorAccess is our access point to the services provided by
      *                      ThinAir Server.
      * @param applicationLog is used for logging.  It is not utilized in this
      *                      sample
      */
    public void init(String name, String p, Properties iniProps, ConnectorAccess ca,
        ApplicationLog al)
    {
    //Since we do not use any of the above in this sample, this area is blank
    }



    /**
     * getDevices() is called once by the ThinAir Server during start-up.  It
     * allows a Connector to indicate the types of devices it supports.
     * getDevices() returns an array containing the names of all DeviceProfiles
     * supported by this Connector.  These names are the friendly names used to
     * uniquely identify every DeviceProfile.  To get the friendly name of a
     * particular device, refer to the ThinAir Server Developer Guide or call
     * DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the Device
     * Detective sample Connector and the ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the
     *         devices this Connector supports.
     */
     public String[] getDevices()
    {
        String[] devices = {"TA_WAP","TA_HTML"};
        return devices;
    }



    /**
     * The handle method implements the core logic of a Connector.  It takes an
     * incoming request from a particular device, and returns an appropriate
     * response. This method is called whenever the server receives a request
     * from a type of device that the Connector indicates it supports, destined
     * (as indicated in the request URL) for a specific application. It is the
     * responsibility of the Connector to interpret the request and generate an
     * appropriate response.
     *
     * The server will pass a Device object containing as much information as
     * possible into this method. The Connector can then utilize the particular
     * Device class to determine more detailed information on the capabilities
     * of the  particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request
     *             parameters from the device.
     * @param device a Device object created in the image of the actual device
     *             making this request.
     * @param result a reference to the OutputStream that will be returned to
     *             the device.
     */
    public void handle(Properties reqProps, Device device, OutputStream out) throws       ↙
        IOException
    {

        //Find out what action the user is trying to perform
        String action = reqProps.getProperty("action");
        String output = null;
        String message = null;

        //If this is the first time entry , then action would be null.
        //We then produce a welcome screen
        if (action == null)
        {
            //Detect what type of device the user has
            if (device instanceof WAPDevice)
            {

            //Since this is a WAP device, generate WML.  Below is the WML and how the WML tag↙
                libraries are used to generate it

            //<wml><card id="main" title="Welcome"><p>Welcome to the Database Sample App<br/>↙
                Please choose from the following...</br>
            //<anchor> <go href = "./DBconnector?action=1">Select statement 1</go></anchor></↙
                br><anchor><go href = "./DBconnector?action=2">Select statement 2</go></     ↙
                anchor>
            //</p></card></wml>
            WMLTagDocument deck = new WMLTagDocument();
            DisplayCard card = new DisplayCard("Welcome");
            com.thinairapps.tag.wml.Paragraph p = new com.thinairapps.tag.wml.Paragraph(com. ↙
                thinairapps.tag.wml.Paragraph.ALIGN_LEFT, com.thinairapps.tag.wml.Paragraph. ↙
                MODE_WRAP);

            p.addChild(new com.thinairapps.tag.wml.Text("<b>Welcome to the Database Sample   ↙
                App</b>"));
            p.addChild(new com.thinairapps.tag.wml.Break());
            p.addChild(new com.thinairapps.tag.wml.Text("Please choose from the following    ↙
                ..."));
            p.addChild(new com.thinairapps.tag.wml.Break());

            com.thinairapps.tag.wml.Go go = new com.thinairapps.tag.wml.Go("./DBconnector?    ↙
                action=1", true, com.thinairapps.tag.wml.Go.METHOD_GET);
            com.thinairapps.tag.wml.Anchor anchor = new com.thinairapps.tag.wml.Anchor(go,new↙
                com.thinairapps.tag.wml.Text("Query 1"));

            p.addChild(anchor);
            p.addChild(new com.thinairapps.tag.wml.Break());

            go = new com.thinairapps.tag.wml.Go("./DBconnector?action=2", true, Go.          ↙
                METHOD_GET);
            anchor = new com.thinairapps.tag.wml.Anchor(go, new com.thinairapps.tag.wml.Text ↙
                ("Query 2"));

            p.addChild(anchor);
            p.addChild(new com.thinairapps.tag.wml.Break());
            card.addParagraph(p);

            deck.addCard(card);

            //render the card and then output
            out.write(deck.render().getBytes());
            }
            else
            {
            //If it is not WML, then it can only be HTML.  Below is the actual HTML and how    ↙
                the HTML tag libraries are used to generate it

            //<html><head>Welcome to the Database Sample App</head><body><p>Please choose      ↙
                from the following...</p>
            //<a href ="./DBConnector?action=1">Select statement 1</a><a href = "./            ↙
                DBConnector?action=2">Select statement 2</a>
            //</body></html>
            HTMLTagDocument doc = new HTMLTagDocument();
            com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
            head.addChild (new com.thinairapps.tag.html.Text("Welcome to the Database Sample ↙
                App"));
            doc.setHead(head);

            Body mBody = new Body();

            com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph ↙
                ();
            para.addChild(new com.thinairapps.tag.html.Text("Please choose from the following↙
                ..."));

            mBody.addChild(para);

            com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Select↙
                statement 1", "./DBconnector?action=1", new com.thinairapps.tag.html.Text    ↙
                ("Query 1"));

            mBody.addChild(an1);
            mBody.addChild(new com.thinairapps.tag.html.Break());

            com.thinairapps.tag.html.Anchor an2 = new com.thinairapps.tag.html.Anchor("Select↙
                statement 2", "./DBconnector?action=2", new com.thinairapps.tag.html.Text    ↙
                ("Query  2"));

            mBody.addChild(an2);
            mBody.addChild(new com.thinairapps.tag.html.Break());

            doc.setBody(mBody);

            //render the card and then output
            out.write(doc.render().getBytes());
            }
        }
        else
        //else, the action parameter is populated
        //Open the database connection
        {
            Connection con = null;
            Statement stmt = null;

            ResultSet result = null;
            ResultSetMetaData resultmeta;
            String header1 = null;
            String header2 = null;
            //open connection to DB
            try
            {
                //Using the Microsoft JDBC ODBC Driver
                Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");

                //here's the sun driver
                //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }

            catch (Exception e)
            {
                System.out.println("Failed to load JDBC/ODBC driver.");
                return;
            }

            //Be sure to establish a ODBC connection with Northwind as the DSN
            String URL = "jdbc:odbc:Northwind";

            //there is no username or password
            String username = "";
            String password = "";
            try
            {
                //try to establish connection
                con = DriverManager.getConnection (URL,username,password);

                //create a Statment
                stmt = con.createStatement();
            }
            catch (Exception e)
            {
                System.err.println("problems connecting to "+ URL);
            }


            String query = null;

            //Determine which select statement to use
            if (action.equals("1"))
                query = "Select CompanyName, Phone from Shippers;";
            else
                //action equals 2 so use second SQL
                query = "SELECT DISTINCTROW TOP 10 Products.ProductName, Products.UnitPrice  ↙
                    FROM Products ORDER BY Products.UnitPrice DESC;";

            try
            {
                //execute the query
                result = stmt.executeQuery(query);

                //get metadata
                resultmeta = result.getMetaData();

                //get the first column label, the return is limited to 2 columns due to the  ↙
                    SQL query
                header1 = resultmeta.getColumnLabel(1);
                //get the second column label
                header2 = resultmeta.getColumnLabel(2);
            }
            catch (Exception e)
            {
                String emesg = e.getMessage();
                System.err.println(emesg);
            }


            //retrieve data, generate page depending on client
            if (device instanceof WAPDevice)
                {

                    //Generate result page, using the metadata above to generate the header ↙
                        columns

                    //<wml><card id="result" title="Results"><p><table columns="2"><tr><td>  ↙
                        Header</td><td>Header2</td></tr>
                    //<tr><td>Details</td><td><Details2</td></tr>....</table></p></card></wml↙
                        >
                    WMLTagDocument deck = new WMLTagDocument();
                    DisplayCard card = new DisplayCard("Results");
                    com.thinairapps.tag.wml.Paragraph p = new com.thinairapps.tag.wml.        ↙
                        Paragraph(com.thinairapps.tag.wml.Paragraph.ALIGN_LEFT, com.          ↙
                        thinairapps.tag.wml.Paragraph.MODE_WRAP);
                    p.addChild(new com.thinairapps.tag.wml.Text("<b>Results</b>"));
                    p.addChild(new com.thinairapps.tag.wml.Break());
                    com.thinairapps.tag.wml.Table resultTable = new com.thinairapps.tag.wml. ↙
                        Table("Results", com.thinairapps.tag.wml.Table.ALIGN_CENTER, 2);
                    com.thinairapps.tag.wml.TableRow tr = new com.thinairapps.tag.wml.        ↙
                        TableRow();
                    com.thinairapps.tag.wml.TableCell tc1 = new com.thinairapps.tag.wml.      ↙
                        TableCell();
                    com.thinairapps.tag.wml.TableCell tc2 = new com.thinairapps.tag.wml.      ↙
                        TableCell();

                    tc1.addChild(new com.thinairapps.tag.wml.Text(header1));
                    tc2.addChild(new com.thinairapps.tag.wml.Text(header2));
                    tr.addChild(tc1);
                    tr.addChild(tc2);
                    resultTable.addChild(tr);
                    try
                    {
                        String detail1;
                        String detail2;

                        //now go through the result set and render the table until result is ↙
                            empty
                        while(result.next())
                        {
                        detail1 = result.getString(1);
                        detail2 = result.getString(2);
                        tr = new com.thinairapps.tag.wml.TableRow();
                        tc1 = new com.thinairapps.tag.wml.TableCell();
                        tc2 = new com.thinairapps.tag.wml.TableCell();
                        tc1.addChild(new com.thinairapps.tag.wml.Text(detail1));
                        tc2.addChild(new com.thinairapps.tag.wml.Text(detail2));
                        tr.addChild(tc1);
                        tr.addChild(tc2);
                        resultTable.addChild(tr);
                        }
                    }
                    catch (Exception e)
                    {
                        String emesg = e.getMessage();
                        System.err.println(emesg);
                    }
                    p.addChild(resultTable);
                    card.addParagraph(p);
                    deck.addCard(card);

                    //render the deck and output the result
                    out.write(deck.render().getBytes());

                    }
                    else

                        if (device instanceof HTMLDevice)
                        {
                            //Below is the result page, first the HTML source , then how the ↙
                                HTML Tag libraries are used
                            //<html><head>Here's the result</head><body><table><tr><td>      ↙
                                Header1</td><td>Header2</td></tr>
                            //<tr><td>detail1</td><td>detail2</td></tr>....</table>
                            //</body></html>

                        HTMLTagDocument doc = new HTMLTagDocument();
                        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.   ↙
                            Head();
                        head.addChild (new com.thinairapps.tag.html.Text("Results"));
                        doc.setHead(head);
                        Body mainbody = new Body();
                        com.thinairapps.tag.html.Paragraph mainpara = new com.thinairapps.tag↙
                            .html.Paragraph();

                        com.thinairapps.tag.html.Table resulttable = new com.thinairapps.tag.↙
                            html.Table(1);
                        com.thinairapps.tag.html.TableRow tr = new com.thinairapps.tag.html. ↙
                            TableRow();
                        com.thinairapps.tag.html.TableCell tc1 = new com.thinairapps.tag.html↙
                            .TableCell();
                        com.thinairapps.tag.html.TableCell tc2 = new com.thinairapps.tag.html↙
                            .TableCell();
                        tc1.addChild(new com.thinairapps.tag.html.Text(header1));
                        tc2.addChild(new com.thinairapps.tag.html.Text(header2));
                        tr.addChild(tc1);
                        tr.addChild(tc2);
                        resulttable.addChild(tr);
                        try
                        {
                            String detail1;
                            String detail2;

                            //run through the result set and render the table, until result  ↙
                                is empty
                            while(result.next())
                            {
                                detail1 = result.getString(1);
                                detail2 = result.getString(2);
                                tr = new com.thinairapps.tag.html.TableRow();
                                tc1 = new com.thinairapps.tag.html.TableCell();
                                tc2 = new com.thinairapps.tag.html.TableCell();
                                tc1.addChild(new com.thinairapps.tag.html.Text(detail1));
                                tc2.addChild(new com.thinairapps.tag.html.Text(detail2));
                                tr.addChild(tc1);
                                tr.addChild(tc2);
                                resulttable.addChild(tr);
                            }

                        }
                        catch (Exception e)
                        {
                            String emesg = e.getMessage();
                            System.err.println(emesg);
                        }
                        mainpara.addChild(resulttable);
                        mainbody.addChild(mainpara);
                        doc.setBody(mainbody);

                        //render the document and output the result
                        out.write(doc.render().getBytes());

                        }

                //close the result, stmt and con objects
                try
                {
                    result.close();
                    stmt.close();
                    con.close();
                }
                catch (Exception e)
                {
                    System.err.println("Error on closing");
                }
            }
        }
    }
```

```
//Core ThinAir Server API functionality
import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.exception.*;

//Core Java API
import java.util.*;

public class TestProviderContext extends StoreProviderContext
{
    public static final short RESULT = 888;

    // Version information
    protected static final String VERSION      = "1.0";
    protected static String APP_NAME           = "TestProvider";
    protected static final String MANUF_NAME   = "ThinAirApps";
    protected static final String MANUF_CONT   = "www.ThinAirApps.com";
    protected static final String BUILD        = "1";
    protected static final Date   APP_RELEASED = new Date ();

    public StoreProviderType getType()
    {
        //Not used by this Provider
        return null;
    }

    public StoreProviderInfo getInfo ()
    {
        return new StoreProviderInfo ( MANUF_NAME,
                                       MANUF_CONT,
                                       APP_NAME,
                                       VERSION,
                                       BUILD,
                                       APP_RELEASED );
    }
}
```

```java
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.device.*;

import com.thinairapps.tag.html.*;
import com.thinairapps.tag.wml.*;

import java.io.*;
import java.util.*;
import java.sql.*;

public class TestProviderConnector implements Connector
{
    private ConnectorAccess myCA;
    private String connectorName;
    private String path;

    public void init(String name, String path, Properties iniProps, ConnectorAccess ca, com. ↙
        thinairapps.platform.connector.ApplicationLog appLog)
    {
        connectorName = name;
        myCA = ca;
        this.path = path;
    }

    public String[] getDevices()
    {
        String[] devices = {"TA_WAP","TA_HTML"};
        return devices;
    }

    public void handle(Properties reqProps, Device device, OutputStream out) throws          ↙
        IOException
    {
        //Find out what action the user is trying to perform
        String action = reqProps.getProperty("action");
        String output = null;
        String message = null;

        String sessionID = null;

        //If this is the first time entry , then action would be null.
        //We then produce a welcome screen
        if (action == null)
        {
            //Detect what type of device the user has
            if (device instanceof WAPDevice)
            {

            //Since this is a WAP device, generate WML.  Below is the WML and how the WML tag↙
                libraries are used to generate it

            //<wml><card id="main" title="Welcome"><p>Welcome to the Database Sample App<br/>↙
                Please choose from the following...</br>
            //<anchor> <go href = "./DBconnector?action=1">Select statement 1</go></anchor></↙
                br><anchor><go href = "./DBconnector?action=2">Select statement 2</go></     ↙
                anchor>
            //</p></card></wml>
            WMLTagDocument deck = new WMLTagDocument();
            DisplayCard card = new DisplayCard("Welcome");
            com.thinairapps.tag.wml.Paragraph p = new com.thinairapps.tag.wml.Paragraph(com. ↙
                thinairapps.tag.wml.Paragraph.ALIGN_LEFT, com.thinairapps.tag.wml.Paragraph. ↙
                MODE_WRAP);

            p.addChild(new com.thinairapps.tag.wml.Text("<b>Welcome to the Database Sample    ↙
                App</b>"));
            p.addChild(new com.thinairapps.tag.wml.Break());
            p.addChild(new com.thinairapps.tag.wml.Text("Please choose from the following    ↙
                ..."));
        p.addChild(new com.thinairapps.tag.wml.Break());

        com.thinairapps.tag.wml.Go go = new com.thinairapps.tag.wml.Go("./DBconnector?   ↙
            action=1", true, com.thinairapps.tag.wml.Go.METHOD_GET);
        com.thinairapps.tag.wml.Anchor anchor = new com.thinairapps.tag.wml.Anchor(go,new↙
            com.thinairapps.tag.wml.Text("Query 1"));

        p.addChild(anchor);
        p.addChild(new com.thinairapps.tag.wml.Break());

        go = new com.thinairapps.tag.wml.Go("./DBconnector?action=2", true, Go.          ↙
            METHOD_GET);
        anchor = new com.thinairapps.tag.wml.Anchor(go, new com.thinairapps.tag.wml.Text ↙
            ("Query 2"));

        p.addChild(anchor);
        p.addChild(new com.thinairapps.tag.wml.Break());
        card.addParagraph(p);

        deck.addCard(card);

        //render the card and then output
        out.write(deck.render().getBytes());
        }
        else
        {
        //If it is not WML, then it can only be HTML.  Below is the actual HTML and how  ↙
            the HTML tag libraries are used to generate it

        //<html><head>Welcome to the Database Sample App</head><body><p>Please choose    ↙
            from the following...</p>
        //<a href ="./DBConnector?action=1">Select statement 1</a><a href = "./          ↙
            DBConnector?action=2">Select statement 2</a>
        //</body></html>
        HTMLTagDocument doc = new HTMLTagDocument();
        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
        head.addChild (new com.thinairapps.tag.html.Text("Welcome to the Database Sample ↙
            App"));
        doc.setHead(head);

        Body mBody = new Body();

        com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph ↙
            ();
        para.addChild(new com.thinairapps.tag.html.Text("Please choose from the following↙
            ..."));

        mBody.addChild(para);

        com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Select↙
            statement 1", path + "?action=1", new com.thinairapps.tag.html.Text("Query   ↙
            1"));

        mBody.addChild(an1);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        com.thinairapps.tag.html.Anchor an2 = new com.thinairapps.tag.html.Anchor("Select↙
            statement 2", path + "?action=2", new com.thinairapps.tag.html.Text("Query   ↙
            2"));

        mBody.addChild(an2);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        doc.setBody(mBody);

        //render the card and then output
        out.write(doc.render().getBytes());
        }
    }
```

```
        else
        //else, the action parameter is populated
        //get the Provider Session
        {

            ResultSet result;
            try
            {
            //Create a session for this user
            //The session ID will be passed back and forth in the request URL
            sessionID = myCA.createProviderSession(TestProviderContext.APP_NAME);
            //reqProps.put("sid", sessionID);

            initProvider(sessionID);
            }
            catch(Exception e)
            {
                //catch initProvider exception here
            }

            //after init, retrieve data
            result = getResultSet(sessionID, action);


            ResultSetMetaData resultmeta;
            String header1 = null;
            String header2 = null;

            try
            {
            //get metadata from result
            resultmeta = result.getMetaData();

            //get the first column label, the return is limited to 2 columns due to the SQL ↙
                query
            header1 = resultmeta.getColumnLabel(1);

            //get the second column label
            header2 = resultmeta.getColumnLabel(2);

            }
            catch (Exception e)
            {
                String emesg = e.getMessage();
                System.err.println(emesg);
            }


            //retrieve data, generate page depending on client
            if (device instanceof WAPDevice)
                {

                    //Generate result page, using the metadata above to generate the header ↙
                        columns

                    //<wml><card id="result" title="Results"><p><table columns="2"><tr><td> ↙
                        Header</td><td>Header2</td></tr>
                    //<tr><td>Details</td><td><Details2</td></tr>....</table></p></card></wml↙
                        >
                    WMLTagDocument deck = new WMLTagDocument();
                    DisplayCard card = new DisplayCard("Results");
                    com.thinairapps.tag.wml.Paragraph p = new com.thinairapps.tag.wml.     ↙
                        Paragraph(com.thinairapps.tag.wml.Paragraph.ALIGN_LEFT, com.       ↙
                        thinairapps.tag.wml.Paragraph.MODE_WRAP);
                    p.addChild(new com.thinairapps.tag.wml.Text("<b>Results</b>"));
                    p.addChild(new com.thinairapps.tag.wml.Break());
                    com.thinairapps.tag.wml.Table resultTable = new com.thinairapps.tag.wml. ↙
                        Table("Results", com.thinairapps.tag.wml.Table.ALIGN_CENTER, 2);
                    com.thinairapps.tag.wml.TableRow tr = new com.thinairapps.tag.wml.    ↙
                        TableRow();
                    com.thinairapps.tag.wml.TableCell tc1 = new com.thinairapps.tag.wml.  ↙
                        TableCell();
                    com.thinairapps.tag.wml.TableCell tc2 = new com.thinairapps.tag.wml.  ↙
                        TableCell();

                    tc1.addChild(new com.thinairapps.tag.wml.Text(header1));
                    tc2.addChild(new com.thinairapps.tag.wml.Text(header2));
                    tr.addChild(tc1);
                    tr.addChild(tc2);
                    resultTable.addChild(tr);
                    try
                    {
                        String detail1;
                        String detail2;

                        //now go through the result set and render the table until result is ↙
                            empty
                        while(result.next())
                        {
                        detail1 = result.getString(1);
                        detail2 = result.getString(2);
                        tr = new com.thinairapps.tag.wml.TableRow();
                        tc1 = new com.thinairapps.tag.wml.TableCell();
                        tc2 = new com.thinairapps.tag.wml.TableCell();
                        tc1.addChild(new com.thinairapps.tag.wml.Text(detail1));
                        tc2.addChild(new com.thinairapps.tag.wml.Text(detail2));
                        tr.addChild(tc1);
                        tr.addChild(tc2);
                        resultTable.addChild(tr);
                        }
                    }
                    catch (Exception e)
                    {
                        String emesg = e.getMessage();
                        System.err.println(emesg);
                    }


                    p.addChild(resultTable);
                    card.addParagraph(p);
                    deck.addCard(card);

                    //render the deck and output the result
                    out.write(deck.render().getBytes());

                    }
                    else

                        if (device instanceof HTMLDevice)
                        {
                            //Below is the result page, first the HTML source , then how the ↙
                                HTML Tag libraries are used
                            //<html><head>Here's the result</head><body><table><tr><td>      ↙
                                Header1</td><td>Header2</td></tr>
                            //<tr><td>detail1</td><td>detail2</td></tr>....</table>
                            //</body></html>

                        HTMLTagDocument doc = new HTMLTagDocument();
                        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.    ↙
                            Head();
                        head.addChild (new com.thinairapps.tag.html.Text("Results"));
                        doc.setHead(head);
                        Body mainbody = new Body();
                        com.thinairapps.tag.html.Paragraph mainpara = new com.thinairapps.tag↙
                            .html.Paragraph();

                        com.thinairapps.tag.html.Table resulttable = new com.thinairapps.tag.↙
```

09874360.060401

```
                            html.Table(1);
                        com.thinairapps.tag.html.TableRow tr = new com.thinairapps.tag.html. ↙
                            TableRow();
                        com.thinairapps.tag.html.TableCell tc1 = new com.thinairapps.tag.html↙
                            .TableCell();
                        com.thinairapps.tag.html.TableCell tc2 = new com.thinairapps.tag.html↙
                            .TableCell();
                        tc1.addChild(new com.thinairapps.tag.html.Text(header1));
                        tc2.addChild(new com.thinairapps.tag.html.Text(header2));
                        tr.addChild(tc1);
                        tr.addChild(tc2);
                        resulttable.addChild(tr);
                        try
                        {
                            String detail1;
                            String detail2;

                            //run through the result set and render the table, until result  ↙
                                is empty
                            while(result.next())
                            {
                                detail1 = result.getString(1);
                                detail2 = result.getString(2);
                                tr = new com.thinairapps.tag.html.TableRow();
                                tc1 = new com.thinairapps.tag.html.TableCell();
                                tc2 = new com.thinairapps.tag.html.TableCell();
                                tc1.addChild(new com.thinairapps.tag.html.Text(detail1));
                                tc2.addChild(new com.thinairapps.tag.html.Text(detail2));
                                tr.addChild(tc1);
                                tr.addChild(tc2);
                                resulttable.addChild(tr);
                            }

                        }
                        catch (Exception e)
                        {
                            String emesg = e.getMessage();
                            System.err.println(emesg);
                        }

                        mainpara.addChild(resulttable);
                        mainbody.addChild(mainpara);
                        doc.setBody(mainbody);

                        //render the document and output the result
                        out.write(doc.render().getBytes());

                        }
            }

    }

    private void initProvider(String sessionID) throws Exception
    {
        //Retrieve the StoreProviderProxy for an existing session
        StoreProviderProxy spProxy = myCA.getStoreProvider(sessionID);

        //Construct this object for pedagogical purposes only
        StoreProviderLogin login = new StoreProviderLogin(null, null, null);
        SupportedItems supports = spProxy.connectUser(login);

        if (supports == null)
            throw new Exception("Error in connectUser.  Provider is unavailable");
    }

    private ResultSet getResultSet(String sessionID, String action)
    {
        ProviderTestResult result = null;
        try
        {
        //Retrieve the StoreProviderProxy for an existing session
        StoreProviderProxy spProxy = myCA.getStoreProvider(sessionID);

        //Construct the request object
        UserDataRequest request = new UserDataRequest();
        request.requests = new ItemRequest[1];
        request.requests[0] = new ItemRequest();
        //request.requests[0].itemType = SimpleWebProviderContext.WEB_PROVIDER_RESULT;
        request.requests[0].bounds = new Bound[1];
        request.requests[0].bounds[0] = new ActionBound(action);

        //Contact the Provider and make the UserDataRequest
        UserData response = spProxy.getUserData(request);


        //Extract the data
        StoreItems items = response.responses[0].items;
        result = (ProviderTestResult) items.elementAt(0);
        }

        catch (Exception e)
        {
            //Catch all exceptions and generate an error page
            e.printStackTrace();

        }
        return result.getResultSet();
    }
}
```

```
import com.thinairapps.platform.provider.*;

import java.util.*;
import java.io.*;
import java.sql.*;

public class TestProvider implements StoreProvider
{
    Connection con = null;

    public SupportedItems connectUser(StoreProviderLogin login, StoreProviderContext context)
    {
        //Create a connection object here to connect to the actual database, if we use
        //an actual login, we use the StoreProviderLogin to get the data
        //returns nothing since we don't have any thing for supported items
        //perhaps we can do read or write as supported.  Normally in groupware it would be
        //messages, contacts, calendar, etc...

            SupportedItems supportedItems;

            //open connection to DB
            try
            {
                //Using the Microsoft JDBC ODBC Driver
                Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");

                //here's the sun driver
                //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }

            catch (Exception e)
            {
                System.out.println("Failed to load JDBC/ODBC driver.");
                return null;
            }

            //Be sure to establish a ODBC connection with Northwind as the DSN
            String URL = "jdbc:odbc:Northwind";

            //there is no username or password

            //retrieve login and password from the StoreProviderLogin Object
            String username = login.name;
            String password = login.password;
            try
            {
                //try to establish connection
                con = DriverManager.getConnection (URL,username,password);
            }
            catch(Exception e)
            {
            }

            //need to return a SupportedItems object, even though we do not use that in this
            //sample application

            // Support no actions
            supportedItems = new SupportedItems();
            String name = TestProviderContext.APP_NAME;
            String location = "none";
            short actions[] = new short[0];
            SupportedItem item = new SupportedItem(TestProviderContext.RESULT, name, location↙
                , actions);
            supportedItems.addItem(item);

            return supportedItems;
    }

    public void disconnectUser()
    {
        try
        {
            con.close();
        }
        catch (Exception e)
        {
            System.err.println("Error on closing");
        }
        con = null;
        //disconnectUser, disconnect from the DB, set all connections to NULL
    }

    public UserDataActionResponse doUserDataAction(UserDataAction action)
    {   return null;
        //This is the WRITE action.  So the same here..send in SQL in the UserDataAction? or
        //maybe send in a number which correlates to a SQL internally in backend
    }

    public UserDataLocations getLocations(UserDataLocationRequest req)
    {   return null;
        //No locations for this app..perhaps we can specify a database for a location...hmmmm
    }

    public UserData getUserData(UserDataRequest request)
    {
        //Get the actual request, which is in the UserDataRequest(request object)
        ItemRequest itemReq = request.requests[0];

        // Prepare the return object
        UserData ud = new UserData();
        //Create one ItemRequestResponse array entry(Why does it need to be an array?)
        ud.responses = new ItemRequestResponse[ 1 ];
        //Now actual populate the 1st index with an ItemRequestResponse object
        ud.responses[0] = new ItemRequestResponse();
        //Set the request field to the itemReq (the UserDataRequest object that was passed in↙
            the parameters)
        ud.responses[0].request = itemReq;

        //In the above, why do we need to store the actual itemReq, why not just use it?

        //What is the reason for the below?
        //short type = itemReq.itemType;

        // Verify that the request is of the correct itemType
        //must change
        //if (type != /*SimpleWebProviderContext.WEB_PROVIDER_RESULT*/)
        //    throw new RuntimeException("Unknown item request type: "+type);

        // The requested URL is wrapped in a StringBound

        //Why put it into a bound object?  Unless Bound is the actual request?

        StringBound bound = (StringBound) ud.responses[0].request.bounds[0];
        String action = (String) bound.getValue();

        //get the actual data..the READ portion...wrap a SQL statement or a numeric
        //represnetation of a SQL statement in the UserDataRequest

        Statement stmt = null;

        ResultSet result = null;

        try
        {
            //create a Statment
            stmt = con.createStatement();
        }
        catch (Exception e)
        {
            System.err.println("problems connecting to database");
        }

        String query = null;

        //Determine which select statement to use
        //retrieve using the UserDataRequest object
        if (action.equals("1"))
            query = "Select CompanyName, Phone from Shippers;";
        else
            //action equals 2 so use second SQL
            query = "SELECT DISTINCTROW TOP 10 Products.ProductName, Products.UnitPrice FROM ↙
                Products ORDER BY Products.UnitPrice DESC;";

        try
        {
                //execute the query
            result = stmt.executeQuery(query);
        }
        catch (Exception e)
        {
            String emesg = e.getMessage();
            System.err.println(emesg);
        }



        ProviderTestResult returnresult = new ProviderTestResult(result);
        //return the resultset and resultmeta data back to the connector
        // Finish loading the return object
        ud.responses[0].items = new StoreItems();
        ud.responses[0].items.addElement(returnresult);

        //close statement
        //try
        //{
        //stmt.close();
        //}
        //catch(Exception e)
        //{
        //}

        return ud;
    }

}
```

```
//Core ThinAir Server API functionality
import com.thinairapps.platform.provider.*;


public class ActionBound extends StringBound
{
    public ActionBound(String action)
    {
        super(action, StringBound.COND_EQUALS);
    }
}
```

```
//Core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.provider.*;
import java.util.*;
import java.sql.*;

public class ProviderTestResult extends StoreItem
{
    private ResultSet result;

    public ProviderTestResult(ResultSet r)
    {
        super();
        result = r;

    }

    public ResultSet getResultSet()
    {
    return result;
    }

}
```

```
===============================================================================

                    Wireless Forms Sample Application

                     Wireless SDK for ThinAir Server

===============================================================================
```

```
-----------------
About this Sample
-----------------
```

The goal of this application is to provide an example of an application which interacts with a JDBC-accessible relational database. Forms and Views are displayed in both HTML and WML, allowing the user to update and query data in a remote database from their wireless device.

Wireless Forms Applications are defined in a simple XML document which conforms to the following framework (there is no DTD defined):

```
<application name="">
    <database>
        <dsn></dsn>
        <login></login>
        <password></password>
    </database>
    <views>
        <view name="">
            <query></query>
        </view>
    </views>
    <forms>
        <form name="">
            <query></query>
            <mappings>
                <mapping>
                        <input></input>
                        <field></field>
                </mapping>
             </mappings>
        </form>
    </forms>
</application>
```

Here is an example Application definition:

```
<application name="User Manager">
    <database>
        <dsn>jdbc:odbc:sample_app</dsn>
        <login>user1</login>
        <password>password</password>
    </database>
    <views>
        <view name="Users">
           <query>SELECT login AS Users, password AS Pwd FROM users</query>
        </view>
    </views>
    <forms>
        <form name="New User">
           <query>insert into users (login, password) select '$lgn', '$pwd'</query>
           <mappings>
           <display>
                    <input>UserName</input>
                        <field>lgn</field>
                        <type>text</type>
           </display>
           <display>
                        <input>Password</input>
                        <field>pwd</field>
                        <type>password</type>
           </display>
           </mappings>
        </form>
    </forms>
</application>
```

For the two included sample applications to run you need to register them with an ODBC dsn as follows:

```
sample.mdb    DSN: "sample_app"
northwin.mbd  DSN: "Northwind"

NOTE: The included sample databases are only appropriate for use on
      Microsoft Windows systems.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * devices.jar

It also requires the following jars included with the sample:

    * dom.jar

    * xml4j.jar

------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - sample connector configuration file

    /src - Java sample code

    /bin - compiled Java sample code

-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  The included
MAKE script will compile the sample using the JDK, if available.

Install the Connector classes, the connector.ini configuration file, and the
"Applications" directory with the XML application definitions, into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Make sure all of the databases specifed in the XML application definitions
are accessible. If you are using the sample MS Access databases, make
sure the Data Source Names (DSNs) are configured through the ODBC datasource
manager, available in the Control Panel.

The jars xml4j.jar and dom.jar must be added to your class path.  Edit the file
StartServer.bat to include the following entries:

    Connectors/WirelessForms/xml4j.jar;
    Connectors/WirelessForms/dom.jar;


Start the ThinAir Server.  It should load WirelessFormsConnector, which should
then in turn load all XML application definitions within its defined
"ApplicationDefinitionDirectory" directory.  The directory defined in the
supplied connector.ini is "<thinairserver install directory>\connectors
\wirelessforms\applications" Place the xml files into that directory.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the WirelessFormsConnector
has been loaded and initialized.  From your wireless device or web browser,
enter the IP address of your machine/wforms (or whatever the value for
ApplicationPath is set to in connector.ini above.  For a machine
with IP address 111.222.12.34 this would be:
     http://111.222.12.34/wforms

Supported devices include WAP: phones, HDML phones, Palm Pilots, Windows CE
devices, desktop web browsers, and GO America/GO RIM pagers.  To create a PQA
application for the Palm VII that integrates with the ThinAir Server, you
will need to understand and use "Web Clipping" technology from Palm.  Web
Clipping involves essentially creating HTML interfaces into your applications.
For your convienence, an HTML file (wforms.html) has been provided for
this purpose.  To find out more about creating PQAs and Web Clipping
technology, visit: http://www.palmos.com/dev/tech/webclipping/


===============================================================================
                          Last updated: 11.18.2000

                    Copyright 1999, 2000 ThinAirApps Inc
===============================================================================
/**
 * @(#)WView.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

public class WView
{
    private String name;

    private String key;
    private String key_query;

    private String query;

    public WView (String name, String key, String key_query, String query)
    {
        this.name = name;
        this.key = key;
        this.query = query;
    }

    public String getName()
    {
        return name;
    }

    public String getKey()
    {
        return key;
    }

    public String getQuery()
    {
        return query;
    }
}
```

```
/**
 * @(#)WMLResultSetDeck.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import com.thinairapps.tag.wml.*;
import java.sql.*;

public class WMLResultSetDeck extends WMLTagDocument
{

    /**
     * Returns the result in Table format
     *
     * @param resultSet is a ResultSet from the query results
     *
     */

    public WMLResultSetDeck (ResultSet resultSet) throws SQLException
    {
        super ();

        ResultSetMetaData metaData = resultSet.getMetaData();
        int numberOfColumns =  metaData.getColumnCount();


        //for each row first display primary key

        Card card = new Card("k1","View");
        String value = null;

        Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);
        int keyIdx = 1;
        int rowIdx = 0;
        String cardName = null;

         addCard (card);

         p.addChild(new Bold(metaData.getColumnLabel(1)));
         p.addChild(new Break());
         card.addChild(p);

        int max = 10;

         while (resultSet.next() && rowIdx < max )
         {
             cardName = "r" + rowIdx++;

            value = resultSet.getObject(keyIdx).toString();

            p.addChild(new Anchor(new Go("#" + cardName,false),new Text(value)));

            Card card2 = null;
            String label = null;

            card2 = new Card(cardName);
            Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

            for(int column = 2; column <= numberOfColumns; column++)
            {
                label = metaData.getColumnLabel(column) + ":";
                value = resultSet.getObject(column).toString();
                p2.addChild(new Bold(label));
                p2.addChild(new Text(value));
                p2.addChild(new Break());
            }

            card2.addChild(p2);
            addCard (card2);

        }

    }
}
```

```
/**
 * @(#)WMLApplicationRenderer.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

import java.util.Enumeration;
import java.util.Vector;
import java.net.URLEncoder;

import java.util.Properties;

import java.sql.*;

/**
 * This class implement the ApplicationRenderer interface. See that class for
 * more information on each method.
 */

public class WMLApplicationRenderer implements ApplicationRenderer, ApplicationConstants
{

    /**
     * Render all currently loaded applications in a selectable list
     *
     * @param apps a hashtable of Application objects
     *
     * @return a String with WML tags
     */

    public String renderApplications (java.util.Hashtable apps)
    {
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_NOWRAP);

        p.addChild(new Text("Choose Application:"));
        p.addChild(new Break());

        String action = null;

        Enumeration enum = apps.elements();

        Application app = null;

        String url = null;

        Properties urlProps = new Properties();
        urlProps.put (ACTION_ARG,MENU_ACTION);

        while (enum.hasMoreElements())
        {
            app = (Application)enum.nextElement();
            urlProps.put (APP_ARG,app.getName());

            url = URLBuilder.buildWapUrl ("?",urlProps,true);
            p.addChild(new Anchor(new Go(url,false),new Text(app.getName())));
           p.addChild(new Break());

        }
        WMLTagDocument doc = new WMLTagDocument();

        Card mCard = new Card("wf","Wireless Forms");
        mCard.addChild(p);

        doc.setCard(mCard);

        return doc.render();

    }

    /**
     * Render a menu for a single Application that allows you to select WForms and WViews
     *
     * @param app an Application instance
     *
     * @return String with WML Tags for a menu
     */

    public String renderMenu (Application app)
    {
          Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_NOWRAP);

           p.addChild(new Text("Choose Application:"));
           p.addChild(new Break());

           String action = null;


           Enumeration enum = app.getForms();

           WForm form = null;

           p.addChild(new Bold("FORMS:"));
           p.addChild(new Break());

           String url = null;

           Properties urlProps = new Properties();
           urlProps.put(ACTION_ARG,FORM_ACTION);
           urlProps.put(APP_ARG,app.getName());

           while (enum.hasMoreElements())
           {
               form = (WForm)enum.nextElement();
               urlProps.put(ITEM_ID,form.getName());

               url = URLBuilder.buildWapUrl ("?",urlProps,true);
               p.addChild(new Anchor(new Go(url,false),new Text(form.getName())));
               p.addChild(new Break());

           }

            enum = app.getViews();

           WView view = null;

           p.addChild(new Bold("VIEWS:"));
           p.addChild(new Break());

           urlProps.put (ACTION_ARG,VIEW_ACTION);


           while (enum.hasMoreElements())
           {
               view = (WView)enum.nextElement();

               urlProps.put(ITEM_ID,view.getName());
               url = URLBuilder.buildWapUrl ("?",urlProps,true);
```

```
                p.addChild(new Anchor(new Go(url,false),new Text(view.getName())));
                p.addChild(new Break());

            }

            p.addChild(new Break());


        WMLTagDocument doc = new WMLTagDocument();

        Card mCard = new Card("wf","Wireless Forms");
        mCard.addChild(p);

        doc.setCard(mCard);

        return doc.render();

    }


    /**
     * Render a JDBC ResultSet for a particular Application and a particular WView
     *
     * @param app the application the WView is from
     * @param view the WView the ResultSet was generated from
     * @param resultSet the resultSet generated from a view and its SQL query
     *
     * @return String with WML tags with the result set from the database query
     */

    public String renderView (Application app, WView view, ResultSet resultSet)
    {
        try
        {
            return new WMLResultSetDeck (resultSet).render();
        }
        catch(SQLException se)
        {
            return se.toString();
        }


    }

    /**
     * Render a particular application's form
     *
     * @param app the application the WForm is a part of
     * @param form the WForm to render
     *
     * @return a String with the WML form
     */

    public String renderForm (Application app, WForm form)
    {
        java.util.Properties props = form.getDisplayMap();

        Enumeration keys = props.keys();
        String key = null, label = null;


        java.util.Properties urlP = new java.util.Properties();
        urlP.put ("ap",app.getName());
        urlP.put ("a",INSERT_ACTION);
        urlP.put ("i",form.getName());

        MultipleInputCard mic = new MultipleInputCard ("c1",form.getName());

        LabeledInput[] li = new LabeledInput[props.size()];
        int i = 0;

        while (keys.hasMoreElements())
        {
            key = (String)keys.nextElement();
            label = (String)props.get(key) + ":";
            li[i++] = new LabeledInput(key,label);
            urlP.put(key,"$"+key);
        }

        String url = URLBuilder.buildWapUrl ("?",urlP,true);

        mic.buildCard (url,"Submit",li,Go.METHOD_GET);

        WMLTagDocument deck = new WMLTagDocument();

        deck.addCard(mic);

        return deck.render();

    }

    /**
     * Render a confirmation message with a link to a specific URL
     *
     * @param app the Application the confirmation is for
     * @param title the title to display for the confirmation
     * @param message the confirmation message to display
     * @param url the url to provider a link to
     *
     * @return String with WML tags with confirmation message
     */

    public String renderConfirmation (Application app, String title, String message, String ↙
        url)
    {
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_WRAP);
        p.addChild(new Text(message));

        Properties urlProps = new Properties();
        urlProps.put(ACTION_ARG,APP_ACTION);
        urlProps.put(APP_ARG,app.getName());

        p.addChild(new Anchor(URLBuilder.buildWapUrl("?",urlProps,true),"Ok",new Text      ↙
            ("Ok")));

        WMLTagDocument page = new WMLTagDocument();

        Card card = new Card("c1",title);
        card.addChild(p);
        page.addCard(card);

        return page.render();
    }

}
```

```
/**
 * @(#)WirelessFormsConnector.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

//thinair platform imports
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.device.*;

//standard java imports
import java.util.*;
import java.io.*;

/*
 *
 * This Connector provides wireless clients with a simple
 * user interface for interacting with a relational database
 * or other JDBC/ODBC accessible data.
 */
public class WirelessFormsConnector implements Connector, ApplicationConstants
{

    //the main appplication logic class which this Connector employs
    WirelessForms wf = null;

    //globals to hold information on jdbc drivers and application directory location
    private String DB_DRIVER = null;
    private String APP_DIR = null;

    /**
     * initialize the connector
     * @param name of application
     * @param path to application from URL
     * @param iniProps from the connector.ini file
     * @param ca interface for Connectors to access the server
     * @param al used for logging
     */
    public void init (String appName, String appPath, Properties props, ConnectorAccess ca, ↙
        com.thinairapps.platform.connector.ApplicationLog al)
    {

        //instantiate the central WirelessForms object, shared across all requests
        wf = new WirelessForms(appPath);

        //get the XML application definition directory from provider.ini
        if (props.getProperty("ApplicationDefinitionDirectory") != null)
            APP_DIR = props.getProperty("ApplicationDefinitionDirectory");
        else
            APP_DIR = DEFAULT_APP_DIR;

        //get the JDBC database driver from provider.ini
        if (props.getProperty("DatabaseDriver") != null)
            DB_DRIVER = props.getProperty("DatabaseDriver");
        else
            DB_DRIVER = MS_DB_DRIVER;

        //attempt to initialize WirelessForms object
        try
        {
            wf.init (APP_DIR, DB_DRIVER);
        }
        catch (Exception e)
        {
            System.err.println ("WirelessFormsConnector.init: error on WirelessForms init    ↙
                : " + e);
            e.printStackTrace();
        }
    }

     /**
     * handle an incoming request
     * @param reqProps represents the HTTP request
     * @param device the actual wireless device instance making the request
     * @param out the OutputStream to write back the response
     */
    public void handle (Properties req, Device device, OutputStream out)
    {

        //extract current action using defined variable name constant
        String action = req.getProperty(ACTION_ARG);

        //init object used to store output from renderering
        String output = null;

        //init the renderer superclass
        ApplicationRenderer renderer = null;

        //based on the device type, determine which subclass of
        //ApplicationRenderer to use. Since some WAP devices also
        //supports HTML, we will specifically look for WAP suport first
        if (device instanceof WAPDevice)
        {
            //its a WAP device, so create a WML Renderer
            renderer = new WMLApplicationRenderer ();
        }
        else if (device instanceof HTMLDevice)
        {
            //its a HTML deice, so create an HTML Renderer
            renderer = new HTMLApplicationRenderer ();
        }

        //if the action is NULL or is the default APP_ACTION
        //get the list of available applications
        if (action == null || action.equals(APP_ACTION))
            output = wf.getApplications (renderer);

        //the action tells the server to reload application definitions
        else if (action.equals(RELOAD_ACTION))
        {
            try
            {
                wf.init(APP_DIR, DB_DRIVER);
                output = wf.getApplications (renderer);
            }
            catch (Exception e)
            {
                System.err.println ("WirelessFormsConnector.handle: error on WirelessForms    ↙
                    init: " + e);
            }
        }

        //retrieve and render a Menu, which display Forms and Views, for a specific    ↙
            Application
        else if (action.equals(MENU_ACTION))
            output = wf.getMenu(req.getProperty(APP_ARG), renderer);

        //retrieve and render a View (essentially a JDBC ResultSet)
        else if (action.equals(VIEW_ACTION))
            output = wf.getView(req.getProperty(APP_ARG),req.getProperty(ITEM_ID),req.      ↙
                getProperty(KEY),renderer);

        //
        else if (action.equals(FORM_ACTION))
            output = wf.getForm(req.getProperty(APP_ARG),req.getProperty(ITEM_ID),renderer);

        //insert data into a table, and display a confirmation
        else if (action.equals(INSERT_ACTION))
        {
            output = wf.insertEntry (req.getProperty(APP_ARG),req.getProperty(ITEM_ID),req,  ↙
                renderer);
        }

        //write the output to the OutputStream via a PrintWriter
        PrintWriter ps = new PrintWriter(out);
        ps.println(output);
        ps.flush();
        ps.close();
    }

    /**
     * @return String array containing the names of all DeviceProfiles supported by this    ↙
        Connector.
     *          These names are the friendly names used to uniquely identify every          ↙
        DeviceProfile.
     */
    public String[] getDevices ()
    {
        //This connector will specify three devices: PalmVII, any
        //HTML mini-Browser device and any WAP mini-browser device
        String[] devices = {WAPDeviceProfile.NAME,PalmVIIDeviceProfile.NAME,HTMLDeviceProfile↙
            .NAME};
        return devices;
    }
```

```
/**
 * @(#)WirelessForms.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import org.w3c.dom.*;
import dom.*;

import java.util.*;
import java.sql.*;
import java.net.*;


/**
 * This class contains primary application logic for the WirelessForms
 * application.
 */
public class WirelessForms implements ApplicationConstants
{
    //stores all loaded appications
    private Hashtable apps;

    //stores the JDBC driver to use
    private static String DRIVER = null;

    private static final String DEFAULT_PARSER_NAME = "dom.wrappers.DOMParser";

    private static String m_appPath;

    public WirelessForms (String appPath)
    {
        super ();
        m_appPath = appPath;
    }

    /*
    * build forms and view, store in hashtable
    */
    public void init (String AppDir, String DRIVER) throws Exception
    {
        this.DRIVER = DRIVER;

        apps = new Hashtable();
        Application app = null;

        String[] files = new java.io.File(AppDir).list();

        for (int i = 0; i < files.length; i++)
        {
            app = buildAppFromXML (AppDir + "/" + files[i]);
            apps.put(app.getName(),app);
        }

    }

    /**
     * loads an Application Definition from a URL and creates an Application object
     * from it using a DOM parser
     */
    private static Application buildAppFromXML (String uri) throws Exception
    {
        DOMParserWrapper parser = (DOMParserWrapper)new dom.wrappers.NonValidatingDOMParser ↙
            ();
        Document document = parser.parse(uri);

        String appName = ((Element)document.getElementsByTagName("application").item(0)).   ↙
            getAttribute("name");

        // Get the source info
        Element dbElement = (Element)document.getElementsByTagName ("database").item (0);
        String dsn = getSubNodeValue (dbElement, "dsn");
        String login = getSubNodeValue (dbElement, "login");
        String password = getSubNodeValue (dbElement, "password");

        Application app = new Application(appName,dsn,login,password,m_appPath);

        NodeList viewElements = document.getElementsByTagName ("view");
        Element viewElement = null;

        WView view = null;
        String viewName,key, key_query, query;

        for (int i = 0; i < viewElements.getLength (); i++)
            {

            viewElement = (Element)viewElements.item (i);

            viewName = viewElement.getAttribute("name");

            key = getSubNodeValue (viewElement, "key");
            key_query = getSubNodeValue (viewElement, "key_query");
            query = getSubNodeValue (viewElement, "query");

            view = new WView(viewName,key,key_query,query);
            app.addView(view);
            }

        NodeList formElements = document.getElementsByTagName ("form");
        Element formElement = null;

        WForm form = null;
        String formName, formQuery, dInput, dField;

        Properties dProps = null;

        for (int i = 0; i < formElements.getLength (); i++)
        {

            formElement = (Element)formElements.item (i);

            formName = formElement.getAttribute("name");

            formQuery = getSubNodeValue (formElement, "query");

            dProps = new Properties();
            NodeList dElements = null;
            Element dNode = null;

            Element dMap = (Element)formElement.getElementsByTagName("mappings").item(0);
            dElements = dMap.getElementsByTagName("display");

            for (int n = 0; n < dElements.getLength(); n++)
            {
                dNode = (Element) dElements.item(n);
                dInput = getSubNodeValue (dNode, "input");
                dField = getSubNodeValue (dNode, "field");
                dProps.put (dField, dInput);
            }

            form = new WForm (formName, formQuery, dProps);
            app.addForm (form);

        }

        return app;
    }

    /**
     * gets value from a subnode of the passed element
     */
    private static String getSubNodeValue (Element element,String node)
    {
        try
            {
            return ((Element)element.getElementsByTagName(node).item(0)).getFirstChild().   ↙
                getNodeValue().trim();
            }
        catch (Exception e)
            {
            return null;
            }
    }

    /**
     * open a connection to the database specified by an Application
     * and store it in the Application object
     */
    public void connect (String appName) throws Exception
    {
        Application app = (Application)apps.get(appName);
        app.setConnection(DatabaseTool.openConnection(app.getDSN(),DRIVER,app.getLogin(),app.↙
            getPassword()));
    }

    /*
    * enumerate through apps
    */
    public String getApplications (ApplicationRenderer renderer)
    {
        return renderer.renderApplications(apps);
    }


    /*
    * enumerate through forms and views and build menu
    */
    public String getMenu (String application, ApplicationRenderer renderer)
    {
        Application app = (Application) apps.get(application);
        return renderer.renderMenu(app);
    }

    /*
    * get view name with key value, execute query, display table output
    */
    public String getView (String application, String name, String key, ApplicationRenderer   ↙
        renderer)
    {
        try
        {

        Application app = (Application) apps.get(application);

        if (app.getConnection() == null || app.getConnection().isClosed())
            connect(application);

        WView view = app.getView (name);

        ResultSet resultSet = DatabaseTool.executeSelect (view.getQuery(),app.getConnection   ↙
            ());

        String out = renderer.renderView(app,view,resultSet);

        resultSet.close();

        return out;
        }
        catch(Exception se)
        {
            return se.toString();
        }

    }

    /*
    * build input form from form object
    */
    public String getForm (String application, String name, ApplicationRenderer renderer)
    {
        Application app = (Application)apps.get(application);
        WForm form = app.getForm (name);
        return renderer.renderForm(app,form);
    }

    /*
    * replace vars in insert string, execute insert query, display result
    */
    public String insertEntry (String application, String name, Properties props,        ↙
        ApplicationRenderer renderer)
    {
        Application app = (Application)apps.get(application);

        System.out.println("got app: " + app.getName());
        WForm form = app.getForm (name);

        System.out.println("got form: " + form.getName());

        String url = "/?a=f&i=" + URLEncoder.encode(form.getName()) + "&ap=" + URLEncoder.   ↙
            encode(app.getName());

        try
        {

            if (app.getConnection() == null || app.getConnection().isClosed())
                connect(application);

            Enumeration enum = props.keys();
            String key = null, value = null;

            String query = form.getQuery();

            System.out.println("query: " + query);

            while(enum.hasMoreElements())
            {
                key = (String)enum.nextElement();
                value = props.getProperty(key);
                System.out.println(key + "=" + value);
                query = substitute (query,"$" + key,value);
                System.out.println("updated query: " + query);
            }

            DatabaseTool.executeInsert(query,app.getConnection());

            return renderer.renderConfirmation (app,"Success","Your data was successfully     ↙
                submitted.",url);
        }
        catch(Exception se)
        {
            return renderer.renderConfirmation (app,"Error","There was an error submitting  ↙
                your query: " + se.getMessage(),url);
        }
    }

    /**
     * basic utility app for doing a String substitute
     */
    private static String substitute (String s, String old, String replace)
    {

        int last,first = 0;
        String foo,bar;

        StringBuffer out = new StringBuffer();

        while(s.indexOf(old,first) > 0)
        {
            last = s.indexOf(old,first);
            foo = s.substring(0,last);
            bar = s.substring(last+old.length(),s.length());
            out.append(foo);
            out.append(replace);
            out.append(bar);
            s = out.toString();
            out = new StringBuffer();
            first = foo.length()+replace.length();
        }

        return s;
    }
```

```
/**
 * @(#)HTMLResultSetTable.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import com.thinairapps.tag.html.*;
import java.sql.*;

/**
 * An HTML Tag Lib widget for rendering a JDBC ResultSet in a simple format
 */
public class HTMLResultSetTable extends Table
{

    /**
     * Returns the result in Table format
     *
     * @param resultSet is a ResultSet from the query results
     * @parma borderSize is a int defining the bordersize of the Table
     */
    public HTMLResultSetTable (ResultSet resultSet, int borderSize) throws SQLException
    {
        super (borderSize);

         TableRow tr = new TableRow();

            TableCell cell = null;

            ResultSetMetaData metaData = resultSet.getMetaData();

            int numberOfColumns =  metaData.getColumnCount();

            for(int column = 0; column < numberOfColumns; column++)
            {
                cell = new TableCell();

                cell.addChild(new Bold(metaData.getColumnLabel(column+1)));
                tr.addChild(cell);
            }
            addChild(tr);

            while (resultSet.next())
            {

                tr = new TableRow();

                for (int i = 1; i <= numberOfColumns; i++)
                {
                    cell = new TableCell();
                    cell.addChild(new Text(resultSet.getObject(i).toString()));
                   tr.addChild(cell);
                }

                addChild(tr);
            }

    }

}
```

```
/**
 * @(#)WForm.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import java.util.Properties;

/**
 * The basic WirelessForm form object. Instances of this class are built
 * automatically from the XML Application Definition.
 */
public class WForm
{

    private String name;
    private String query;
    private Properties displayMap;

    /**
     * @param name the displayable form name
     * @param query the insert query to use for submitting the form data
     * @param displayMap a prop mapping form field variables to displayable labels
     */
    public WForm (String name, String query, Properties displayMap)
    {
        this.name = name;
        this.query = query;
        this.displayMap = displayMap;
    }

    public String getName()
    {
        return name;
    }

    public String getQuery()
    {
        return query;
    }

    public Properties getDisplayMap()
    {
        return displayMap;
    }
}
```

```
/**
 * @(#)HTMLApplicationRenderer.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */


//Thinair Tag Library
import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;

//Java Utilities
import java.util.Enumeration;
import java.util.Vector;
import java.net.URLEncoder;

//Java SQL library
import java.sql.*;

/**
* This class implement the ApplicationRenderer interface. See that class for
* more information on each method.
*/

public class HTMLApplicationRenderer implements ApplicationRenderer, ApplicationConstants
{

    /**
     * Render all currently loaded applications in a selectable list
     *
     * @param apps a hashtable of Application objects
     *
     * @return a String with HTML tags
     */

    public String renderApplications (java.util.Hashtable apps)
    {
        try
        {

            Paragraph body = new Paragraph();

            String action = null;

            Enumeration enum = apps.elements();

            Application app = null;

           // body.addChild(new Text("<b>Applications:</b><BR>"));

            com.thinairapps.tag.html.Form hForm = new com.thinairapps.tag.html.Form ↙
                ("f1","","GET");

            Select select = new Select("ap");
            select.addAttribute("size","3");

            while (enum.hasMoreElements())
            {
                app = (Application)enum.nextElement();
                //action = "?a=m&ap=" + URLEncoder.encode(app.getName());
                select.addOption(app.getName(),false);
            }
            hForm.addChild(select);
            hForm.addChild(new Break());

            hForm.addChild(new SubmitButton("Launch"));
            hForm.addChild(new Anchor("Refresh","?a=r"));
            hForm.addChild(new HiddenInput("a","m"));

        HTMLTagDocument doc = new HTMLTagDocument();
        Head head = new Head();
        head.addChild (new Title("Wireless Forms"));
        doc.setHead(head);

        Body mBody = new Body();
        mBody.addChild(new Bold("Select an application:"));
        mBody.addChild(new Break());
        mBody.addChild(hForm);

        doc.setBody(mBody);

        return doc.render();

        }
        catch(InvalidTagException ite)
        {
            return null;
        }
    }



    /**
     * Render a menu for a single Application that allows you to select WForms and WViews
     *
     * @param app an Application instance
     *
     * @return String with HTML Tags for a menu
     */

    public String renderMenu (Application app)
    {

        try
        {

            Paragraph body = new Paragraph();

            Enumeration views = app.getViews();
            WView view = null;
            String action = null;

            com.thinairapps.tag.html.Form viewForm = new com.thinairapps.tag.html.Form
                ("views","","GET");

            viewForm.addChild (new Text("Views: "));
            viewForm.addChild (new HiddenInput ("a","v"));
            viewForm.addChild (new HiddenInput ("ap",app.getName()));
            Select viewSelect = new Select ("i");

            while(views.hasMoreElements())
            {
                view = (WView)views.nextElement();
                viewSelect.addOption (view.getName(),false);
            }

            viewForm.addChild (viewSelect);

            viewForm.addChild (new SubmitButton ("Go"));
            body.addChild (viewForm);
            body.addChild(new Break());

            Enumeration forms = app.getForms();
            WForm form = null;

            //
            com.thinairapps.tag.html.Form formForm = new com.thinairapps.tag.html.Form
                ("forms","","GET");

            formForm.addChild (new Text("Forms: "));
            formForm.addChild (new HiddenInput ("a","f"));
            formForm.addChild (new HiddenInput ("ap",app.getName()));
            Select formSelect = new Select ("i");

            while(forms.hasMoreElements())
                {
                form = (WForm)forms.nextElement();
                formSelect.addOption (form.getName(),false);
                }

            formForm.addChild (formSelect);

            formForm.addChild (new SubmitButton ("Go"));
            body.addChild (formForm);


            HTMLTagDocument doc = new HTMLTagDocument();
            Head head = new Head();
            head.addChild (new Title(app.getName()));
            doc.setHead(head);

            Body mBody = new Body();
            mBody.addChild(body);

            doc.setBody(mBody);

            return doc.render();

        }
        catch (InvalidTagException e)
        {
            return null;
        }


    }

    /**
     * Render a JDBC ResultSet for a particular Application and a particular WView
     *
     * @param app the application the WView is from
     * @param view the WView the ResultSet was generated from
     * @param resultSet the resultSet generated from a view and its SQL query
     *
     * @return String with HTML tags with the result set from the database query
     */

    public String renderView (Application app, WView view, ResultSet resultSet)
    {
        try
        {
            HTMLTagDocument doc = new HTMLTagDocument();

            Head head = new Head();
            head.addChild(new Title(view.getName()));
            doc.setHead(head);

            Body body = new Body();
            body.addChild(new Text("<font size=\"1\" face=\"geneva\">"));

            Table table = new HTMLResultSetTable (resultSet, 1);

            body.addChild(table);
            body.addChild(new Break());

            body.addChild(new HorizontalRule());
            String main = "?a=m&ap=" + URLEncoder.encode(app.getName());
            body.addChild(new Anchor("",main,new Text("[ menu ]")));

            body.addChild(new Text("</font>"));

            doc.setBody(body);
            return doc.render();
        }
        catch(SQLException se)
        {
            return se.toString();
        }
        catch(InvalidTagException se)
        {
            return se.toString();
        }


    }


    /**
     * Render a particular application's form
     *
     * @param app the application the WForm is a part of
     * @param form the WForm to render
     *
     * @return a String with the HTML form
     */
    public String renderForm (Application app, WForm form)
    {
        try
        {

            java.util.Properties props = form.getDisplayMap();

            //build an HTML Form Tag object to render the WForm

            com.thinairapps.tag.html.Form hForm = new com.thinairapps.tag.html.Form("insert",↙
                app.getWFormsURL(),"POST");

            Enumeration keys = props.keys();
            String key = null;

            while (keys.hasMoreElements())
            {
                key = (String)keys.nextElement();
                hForm.addChild(new NonBreakingSpace(4));
                hForm.addChild(new Text("<b>" + (String)props.get(key) + ":</b>&nbsp;"));
                hForm.addChild(new TextField(key));
                hForm.addChild(new Break());
            }

            hForm.addChild(new Break());

            hForm.addFormElement(new HiddenInput(ACTION_ARG,INSERT_ACTION));
            hForm.addFormElement(new HiddenInput(APP_ARG,app.getName()));
            hForm.addFormElement(new HiddenInput(ITEM_ID,form.getName()));

            Center center = new Center();
            center.addChild(new SubmitButton("Submit"));
            center.addChild(new Input("Reset","Reset","Reset"));

            hForm.addChild(center);

            HTMLTagDocument page = new HTMLTagDocument();
            Head head = new Head();
            head.addChild (new Title(form.getName()));
            page.setHead(head);

            Body body = new Body();
            body.addChild(hForm);
            page.setBody(body);

            return page.render();

        }
        catch(InvalidTagException ite)
        {
            return ite.toString();
        }
    }


    /**
     * Render a confirmation message with a link to a specific URL
     *
     * @param app the Application the confirmation is for
     * @param title the title to display for the confirmation
     * @param message the confirmation message to display
     * @param url the url to provider a link to
     *
     * @return String with HTML tags with confirmation message
     */

    public String renderConfirmation (Application app, String title, String message, String ↙
        url)
    {
        try
        {
            Center center = new Center();
            center.addChild(new HorizontalRule());
            center.addChild(new Text(message));

            center.addChild(new HorizontalRule());
            String main = "?a=m&ap=" + URLEncoder.encode(app.getName());
            center.addChild(new Anchor("Ok",main));


            HTMLTagDocument page = new HTMLTagDocument();
            Head head = new Head();
            head.addChild (new Title(title));
            page.setHead(head);

            Body body = new Body();
            body.addChild(center);
            page.setBody(body);

            return page.render();
        }
        catch(InvalidTagException e)
        {
            return e.toString();
        }
    }

}
```

```
/**
 * @(#)DatabaseTool.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import java.sql.*;

/**
 * A utility class that wraps JDBC querying logic
 */

public class DatabaseTool
{

    /**
     * Open a JDBC connection for a particular DSN, username, and password
     */
    public static Connection openConnection(String url, String driverName,String user, String↙
        passwd) throws SQLException, ClassNotFoundException
    {
        Class.forName(driverName);
        return DriverManager.getConnection(url, user, passwd);
    }

    /**
     * execute an SQL  query for the given connection
     */
    public static ResultSet executeSelect (String query, Connection connection) throws       ↙
        SQLException
    {
        Statement statement = connection.createStatement();
        return statement.executeQuery(query);

    }

    /**
     * execute an SQL Insert query for the given connection
     */
    public static boolean executeInsert (String query, Connection connection) throws        ↙
        SQLException
    {
        Statement statement = connection.createStatement();
        return statement.execute(query);
    }

    /**
     * execute a named stored procedure for the given connetion
     */
    public static boolean executeStoredProcedure(String sp,Object[][] params,Connection      ↙
        connection)
    {

        try
        {
            Statement statement = connection.createStatement();
            ResultSet resultSet = null;

            StringBuffer query = new StringBuffer();

            query.append("EXECUTE " + sp + " ");
            for (int i = 0; i < params.length; i++)
            {
                if (params[i][1] instanceof String)
                    query.append("@" + params[i][0] + " = '" + params[i][1] + "', ");
                else if (params[i][1] instanceof Integer)
                    query.append("@" + params[i][0] + " = " + ((Integer)params[i][1]).      ↙
                        intValue() + ", ");
                else if (params[i][1] instanceof Double)
                    query.append("@" + params[i][0] + " = " + ((Double)params[i][1]).       ↙
                        doubleValue() + ", ");
                else if (params[i][1] instanceof java.util.Date)
                    query.append("@" + params[i][0] + " = convert(datetime,'" + ((java.util. ↙
                        Date)params[i][1]).toLocaleString()+ "'), ");
            }

            String command = query.toString();
            command = command.substring(0,command.length()-2);

            System.out.println("executing stored procedure: " + command);

            resultSet = statement.executeQuery(command);

            resultSet.close();
            statement.close();

            //  close(); Need to copy the metaData, bug in jdbc:odbc driver.
            return true;
        }
        catch (SQLException ex)
        {

            System.err.println(ex);
            return false;
        }
    }

    /**
     * closed the passed connectoin
     */
    public static void closeConnection(Connection connection) throws SQLException
    {
        connection.close();
    }
}
```

```
/**
 * @(#)ApplicationConstants.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

public interface ApplicationConstants
{
    //Its important to define short variable names for use in content and URLs
    public final static String ACTION_ARG = "a";
    public final static String APP_ARG = "ap";

    public final static String RELOAD_ACTION = "r";
    public final static String APP_ACTION = "a";
    public final static String MENU_ACTION = "m";
    public final static String VIEW_ACTION = "v";
    public final static String FORM_ACTION = "f";
    public final static String INSERT_ACTION = "i";
    public final static String UPDATE_ACTION = "u";

    public final static String KEY = "k";
    public final static String ITEM_ID = "i";

    //some default values for use in initialization
    public final static String DEFAULT_APP_DIR = "Connectors\\WirelessForms\\apps";
    public final static String SUN_DB_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
    public final static String MS_DB_DRIVER = "com.ms.jdbc.odbc.JdbcOdbcDriver";
}
```

```
/**
 * @(#)Application.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 **
 * This class is the Java representation of the XML DTD used by
 * the Application Definitions Documents. The server parses the
 * XML document to create an instanceof this class per application.
 *
 */

import java.util.Hashtable;
import java.util.Enumeration;

import java.sql.*;

public class Application
{
    //the application name
    private String m_name;

    //the Data Source Name and login and password for the DSN
    private String m_dsn;
    private String m_login;
    private String m_password;
    private String m_wformsURL;

    //the store for all forms defined within an applicatoin
    private Hashtable forms;

    //the store for all views defined within an application
    private Hashtable views;

    //the JDBC Connection object used by each Application
    private Connection conn;

    /**
     * @param name the application name
     * @param dsn the data source name
     * @param login an authorized username for the DSN
     * @param password a corresponding password for the login
     */
    public Application (String name, String dsn, String login, String password, String ↙
        wformsURL)
    {
        m_name = name;
        m_dsn = dsn;
        m_login = login;
        m_password = password;
        m_wformsURL = wformsURL;

        forms = new Hashtable();
        views = new Hashtable();
    }

    /**
     * Set and store a JDBC Connection object within an Application instance
     *
     * @param conn a JDBC Connection object
     */
    public void setConnection (Connection conn)
    {
        this.conn = conn;
    }

    /**
     * Return the current stored JDBC Connection
     *
     * @return a JDBC Connection instance
     */
    public Connection getConnection ()
    {
        return conn;
    }

    public String getName()
    {
        return m_name;
    }

    public String getDSN()
    {
        return m_dsn;
    }

    public String getLogin()
    {
        return m_login;
    }

    public String getPassword()
    {
        return m_password;
    }

    public String getWFormsURL()
    {
        return m_wformsURL;
    }

    public void addForm (WForm form)
    {
        forms.put (form.getName(), form);
    }

    public WForm getForm (String name)
    {
        return (WForm) forms.get(name);
    }

    public Enumeration getForms()
    {
        return forms.elements();
    }

    public void addView (WView view)
    {
        views.put (view.getName(), view);
    }

    public WView getView (String name)
    {
        return (WView) views.get(name);
    }

    public Enumeration getViews()
    {
        return views.elements();
    }
}
```

```
/**
 * @(#)ApplicationRenderer.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

import java.sql.ResultSet;

/**
 * Interface used to build renderers for different markup languages and device types
 *
 */
public interface ApplicationRenderer
{

    /**
     * Render all currently loaded applications in a selectable list
     *
     * @param apps a hashtable of Application objects
     */
    public abstract String renderApplications (java.util.Hashtable apps);

    /**
     * Render a menu for a single Application that allows you to select WForms and WViews
     *
     * @param app an Application instance
     */
    public abstract String renderMenu (Application app);

    /**
     * Render a JDBC ResultSet for a particular Application and a particular WView
     *
     * @param app the application the WView is from
     * @param view the WView the ResultSet was generated from
     * @param resultSet the resultSet generated from a view and its SQL query
     */
    public abstract String renderView (Application app, WView view, ResultSet resultSet);

    /**
     * Render a particular application's form
     * @param app the application the WForm is a part of
     * @param form the WForm to render
     */
    public abstract String renderForm (Application app, WForm form);

    /**
     * Render a confirmation message with a link to a specific URL
     *
     * @param app the Application the confirmation is for
     * @param title the title to display for the confirmation
     * @param message the confirmation message to display
     * @param url the url to provider a link to
     */
    public abstract String renderConfirmation (Application app, String title, String message ↙
        , String url);

}
```

```
==============================================================================

                         Tic Tac Toe Sample Connector

                       Wireless SDK for ThinAir Server

==============================================================================


-----------------
About this Sample
-----------------

This sample connector demonstrates a simple game that can be implemented using
the SDK. The tic-tac-toe game is always between the user and the connector
logic. Alternating games have alternating players starting. All HTML and WML
devices are supported by this connector.

This connector also makes use of session objects. For more information on using
sessions, see the SessionManagement sample connector in this directory and the
corresponding ThinAir Server API documentation.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * devices.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    TicTacToeConnector.jar - compiled Java code

    /src - java source files - ProfileConnector.java and ProfileData.java


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice. Make sure to
append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

In order to display the Images, the GIF files must be placed into a
"tictactoeConnector" directory under htdocs in the ThinAirServer directory.
(ie: /program files/thinairapps/thinairserver/htdocs/tictactoeConnector)

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the Tic Tac Toe Connector has been
loaded and initialized. From your HTML or WML device, enter the IP address
listed as the value for ApplicationPath in connector.ini (your ThinAirServer IP
address), followed by /samples/tictactoe. For a machine with IP address
111.222.12.34 this would be:
     http://111.222.12.34/samples/tictactoe
Follow the on-screen instructions.

===============================================================================
                         Last updated: 11.13.2000
                   Copyright 1999, 2000 ThinAirApps Inc.
===============================================================================
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//core ThinAir Server API functionality
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.exception.*;

//rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;
import com.thinairapps.tag.wml.*;

//Core Java API
import java.util.*;
import java.io.*;



/**
 * This sample Connector demonstrates a simple interactive application that can be created    ↙
     using the
 * ThinAir API. The Connector plays Tic Tac Toe against the user, and works on all HTML and   ↙
     WML
 * devices. The TicTacToeBoard class, contained in TicTacToeBoard.java, contains all the      ↙
     logic for
 * the board itself, and the game rules. The main class, TicTacToeConnector, contains the     ↙
     logic for
 * playing strategy, the game flow, and the screen display.
 */
public class TicTacToeConnector implements Connector {

    //The friendly name of this sample app
    String          appName;
    String          path;
    Properties      props;

    //Our access point to the services of ThinAir Server
    ConnectorAccess access;

    //points to the directory for images
    private final static String IMAGE_PATH = "/docs/tictactoeConnector/";

    Integer GameState;
    final Integer IN_PROGRESS = new Integer(1);
    final Integer USER_WON = new Integer(2);
    final Integer CONNECTOR_WON = new Integer(3);
    final Integer TIE_GAME = new Integer(4);



    /**init() is called by the ThinAirServer when the Connector is loaded.  It provides the ↙
        Connector
     * with resources it needs to interact with the ThinAirServer.
     *
     * @param applicationName is a String derived from connector.ini.  We don't need this for↙
        this sample.
     * @param applicationPath is a String dervid from connector.ini.  We don't need this for ↙
        this sample.
     * @param connectorProps is a Properties list containing developer assigned              ↙
        connector-specific properties.
     * We don't need this parameter in this sample.
     * @param connectorAccess is our access point to the services provided by ThinAir Server ↙
        .  We don't need
     * this for this sample.
     * @param ApplicationLog is used for Logging.  It is not used in this sample
    */
    public void init(String applicationName, String applicationPath, Properties           ↙
        connectorProps,
                     ConnectorAccess connectorAccess, ApplicationLog al)
    {
        appName = applicationName;
        path = applicationPath;
        props = connectorProps;
        access = connectorAccess;
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a     ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing  ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used  ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this      ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String deviceTypes[] = {"TA_HTML", "TA_WAP"};
        return deviceTypes;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming    ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,    ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters   ↙
        from the device.
     * @param device a Device object created in the image of the actual device making this   ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream out) throws IOException ↙
        {

        String resultString;
        this.props = props;

        //we name the sessionID param "sid"
        String sessionID = props.getProperty("sid");
        //the cache for this session
        Hashtable cache = null;

        TicTacToeBoard theBoard = null;
        Integer GameState = null;

        char StartingPlayer, SecondPlayer;

        //if this is the user's first hit, they will not yet have a session
        if (sessionID == null)
        {
            //so create one for them
            sessionID = access.createSession();
        }

        try
        {
            //get the cache for this session...
            cache = access.getSessionCache(sessionID);
        }
        catch (NoSuchSessionException e)
        {
            // do nothing
        }

        if (cache == null)
        {
            // create a new board
            theBoard = new TicTacToeBoard();
            theBoard.init();
            GameState = IN_PROGRESS;
        }
        else if (cache.get("board") == null)
        {
            // create a new board
            theBoard = new TicTacToeBoard();
            theBoard.init();
            GameState = IN_PROGRESS;
        }
        else
        {
            theBoard = (TicTacToeBoard)cache.get("board");
            GameState = (Integer)cache.get("GameState");
        }

        playTurn(sessionID, theBoard);

        cache.put("board", theBoard);

        try {
            // determine which device is contacting the Connector and use a different
            // rendering class to generate output
            if (device instanceof HTMLDevice)
                resultString = HTMLDisplayScreen(sessionID, theBoard);

                    else if (device instanceof PalmVIIDevice)
                        resultString = HTMLDisplayScreen(sessionID, theBoard);

                else if (device instanceof WAPDevice)
                    resultString = WMLDisplayScreen(sessionID, theBoard);

                    else if (device instanceof UPWAPDevice)
                        resultString = WMLDisplayScreen(sessionID, theBoard);

                            else if (device instanceof GoWebRIMDevice)
                    resultString = WMLDisplayScreen(sessionID, theBoard);

            else
                resultString = "ERROR: Device "+device.getClass()+" not supported.";
        } catch (Exception e) {
            e.printStackTrace();
            resultString = "ERROR: "+e.getMessage();
        }

        out.write(resultString.getBytes());
    }



    /**
     * MakeMove() decides on the next move for the computer to make, and changes the board to
     * reflect that move.
     *
     * @param theBoard the current state of the board
     */
    private void makeMove(TicTacToeBoard theBoard)
    {

        Random randGen = new Random();
        int randCell, randRow, randCol;

        int rowCount, colCount;

        //look for a winning move...
        for (rowCount = 0; rowCount < theBoard.NUM_ROWS; rowCount++) {
            for (colCount = 0; colCount < theBoard.NUM_COLS; colCount++) {
                if (theBoard.emptyAt(rowCount, colCount)) {
                    theBoard.placePiece('O', rowCount, colCount);
                    if (theBoard.playerWon('O')) {
                        return;
                    } else {
                        theBoard.placePiece(theBoard.EMPTY_SPACE, rowCount, colCount);
                    }
                }
            }
        }



        //Barring that, block user from any winning moves..
        for (rowCount = 0; rowCount < theBoard.NUM_ROWS; rowCount++) {
            for (colCount = 0; colCount < theBoard.NUM_COLS; colCount++) {
                if (theBoard.emptyAt(rowCount, colCount)) {
                    theBoard.placePiece('X', rowCount, colCount);
                    if (theBoard.playerWon('X')) {
                        theBoard.placePiece('O', rowCount, colCount);
                        return;
                    } else {
                        theBoard.placePiece(theBoard.EMPTY_SPACE, rowCount, colCount);
                    }
                }
            }
        }



        //Otherwise, enter piece in a randomly-chosen cell (of the ones that haven't been
        //filled already
        do {
            randCell = Math.abs(randGen.nextInt() % (theBoard.NUM_COLS * theBoard.NUM_ROWS));
            randRow = randCell / theBoard.NUM_ROWS;
            randCol = randCell % theBoard.NUM_COLS ;
        }
            while (! theBoard.emptyAt(randRow, randCol));

        theBoard.placePiece('O', randRow, randCol);
    }
    /**
     * HTMLDisplayBoard() returns the HTML version of the board in its current state.
     *
     * @param sessionID the current session ID
     * @param theBoard the current state of the board
     * @param GameOver whether or not the game is over and further clicking should be
     * disabled
     *
     * @return an HTML table, containing a graphical representation of the board
     */
    public com.thinairapps.tag.html.Table HTMLDisplayBoard(String sessionID, TicTacToeBoard  ↙
        theBoard,
                                                              boolean GameOver) {

        com.thinairapps.tag.html.Table TTTTable = new com.thinairapps.tag.html.Table(1);
        com.thinairapps.tag.html.TableCell
            Cell[][] = new com.thinairapps.tag.html.TableCell[theBoard.NUM_ROWS][theBoard.  ↙
                NUM_COLS];
        com.thinairapps.tag.html.TableRow
            Row[] = new com.thinairapps.tag.html.TableRow[theBoard.NUM_ROWS];

        int rowCount, colCount;
        char curPiece;

        for (rowCount = 0; rowCount < theBoard.NUM_ROWS; rowCount++) {

            Row[rowCount] = new com.thinairapps.tag.html.TableRow();
            TTTTable.addChild(Row[rowCount]);

            for (colCount = 0; colCount < theBoard.NUM_COLS; colCount++) {

                Cell[rowCount][colCount] = new com.thinairapps.tag.html.TableCell();
                Row[rowCount].addChild(Cell[rowCount][colCount]);


                if (theBoard.emptyAt(rowCount, colCount)) {
                    if (GameOver) {
                        Cell[rowCount][colCount].addChild
                            (new com.thinairapps.tag.html.Image(IMAGE_PATH + "ttt-blank.  ↙
                                gif"));
                    } else {
                        Cell[rowCount][colCount].addChild
                            (new HyperlinkedImage(path + "?row=" + Integer.toString          ↙
                                (rowCount) +
                            "&col=" + Integer.toString(colCount) + "&sid=" + sessionID,
                            IMAGE_PATH + "ttt-blank.gif"));
                    }
                } else {
                    curPiece = theBoard.pieceOccupying(rowCount, colCount);
                    if (curPiece == 'X') {
                        Cell[rowCount][colCount].addChild
                            (new com.thinairapps.tag.html.Image(IMAGE_PATH + "ttt-x.gif"));
                    } else if (curPiece == 'O') {
                        Cell[rowCount][colCount].addChild
                            (new com.thinairapps.tag.html.Image(IMAGE_PATH + "ttt-o.gif"));

                    }
                }
            }
        }
        return TTTTable;
    }


    /**
```

```
     * This method renders an HTML page showing the board in its current state, along with  ↙
       the title,
     * and other explanatory information
     *
     * @return the rendered HTML page.
     */
    private String HTMLDisplayScreen(String sessionID, TicTacToeBoard theBoard)
    {

        String resultString;
        Body body = new Body();
        HTMLTagDocument HTMLDoc = new HTMLTagDocument();
        Title theTitle = new Title("Play Tic Tac Toe!");

        //the cache for this session
        Hashtable cache = null;

        try
        {
            //get the cache for this session...
            cache = access.getSessionCache(sessionID);
        } catch (NoSuchSessionException e) { }

        Character SecondPlayer = (Character)cache.get("SecondPlayer");
        Integer GameState = (Integer)cache.get("GameState");
        boolean GameOver = (GameState == USER_WON || GameState == CONNECTOR_WON || GameState ↙
            == TIE_GAME);

        //set the background color
        body.addAttribute("bgcolor","#ffffff");

        //add a title
        body.addChild(new com.thinairapps.tag.html.Bold("Play a game of Tic Tac Toe"));

        body.addChild(new HorizontalRule());

        body.addChild(HTMLDisplayBoard(sessionID, theBoard, GameOver));

        if (GameState == USER_WON) {
            body.addChild(new com.thinairapps.tag.html.Bold("You win! "));
        } else if (GameState == CONNECTOR_WON) {
            body.addChild(new com.thinairapps.tag.html.Bold("I win! "));
        } else if (GameState == TIE_GAME) {
            body.addChild(new com.thinairapps.tag.html.Bold("It's a tie! "));
        }

        body.addChild(new com.thinairapps.tag.html.Text("Play a "));
        body.addChild(new com.thinairapps.tag.html.Anchor("", path + "?action=clear&sid=" +  ↙
            sessionID,
                                                          new com.thinairapps.tag.html.Text  ↙
                                                            ("new game")));

        HTMLDoc.addChild(theTitle);
        HTMLDoc.addChild(body);

        resultString = HTMLDoc.render();

        return resultString;
    }


    /**
     * WMLDisplayBoard() returns the WML version of the board in its current state.
     *
     * @param sessionID the current session ID
     * @param theBoard the current state of the board
     *
     * @return an HTML table, containing a graphical representation of the board
     */
    public com.thinairapps.tag.wml.Table WMLDisplayBoard(String sessionID, TicTacToeBoard     ↙
        theBoard)
    {

        com.thinairapps.tag.wml.Table
            TTTTable = new com.thinairapps.tag.wml.Table("Board", "ALIGN_LEFT", 3);

        com.thinairapps.tag.wml.TableCell
            Cell[][] = new com.thinairapps.tag.wml.TableCell[theBoard.NUM_ROWS][theBoard.     ↙
                NUM_COLS];

        com.thinairapps.tag.wml.TableRow
            Row[] = new com.thinairapps.tag.wml.TableRow[theBoard.NUM_ROWS];

        int rowCount, colCount;
        char curPiece;

        Random randGen = new Random();

        // append a random number to combat caching
        int randNum = Math.abs(randGen.nextInt() % 10000);

        for (rowCount = 0; rowCount < theBoard.NUM_ROWS; rowCount++) {

            Row[rowCount] = new com.thinairapps.tag.wml.TableRow();
            TTTTable.addChild(Row[rowCount]);

            for (colCount = 0; colCount < theBoard.NUM_COLS; colCount++) {

                Cell[rowCount][colCount] = new com.thinairapps.tag.wml.TableCell();
                Row[rowCount].addChild(Cell[rowCount][colCount]);

                curPiece = theBoard.pieceOccupying(rowCount, colCount);

                if (curPiece == theBoard.EMPTY_SPACE) {
                    Cell[rowCount][colCount].addChild
                        (new com.thinairapps.tag.wml.Image(IMAGE_PATH + "/ttt-blank.     ↙
                            gif", "?"));
                } else if (curPiece == 'X') {
                    Cell[rowCount][colCount].addChild
                        (new com.thinairapps.tag.wml.Image(IMAGE_PATH + "/ttt-x.gif", "X"));
                } else if (curPiece == 'O') {
                    Cell[rowCount][colCount].addChild
                        (new com.thinairapps.tag.wml.Image(IMAGE_PATH + "/ttt-o.gif", "O"));
                }
            }
        }
        return TTTTable;
    }


    /**
     * This method renders a WML page showing the board in its current state, along with the ↙
        title,
     * and other explanatory information
     *
     * @return the rendered WML page.
     */
    private String WMLDisplayScreen(String sessionID, TicTacToeBoard theBoard) {

        String resultString;
        DisplayCard theCard = new DisplayCard("c1");
        WMLTagDocument deck = new WMLTagDocument();

        Random randGen = new Random();
        int randNum = Math.abs(randGen.nextInt() % 10000);
        //the cache for this session
        Hashtable cache = null;

        try
        {
            //get the cache for this session...
            cache = access.getSessionCache(sessionID);
        }
        catch (NoSuchSessionException e) { }

        Integer GameState = (Integer)cache.get("GameState");

        com.thinairapps.tag.wml.Paragraph p = new com.thinairapps.tag.wml.Paragraph();
        p.addChild(WMLDisplayBoard(sessionID, theBoard));

        if (GameState == USER_WON)
        {
            p.addChild(new com.thinairapps.tag.wml.Text("You win! "));
        }
        else if (GameState == CONNECTOR_WON)
        {
            p.addChild(new com.thinairapps.tag.wml.Text("I win! "));
        }
        else if (GameState == TIE_GAME)
        {
            p.addChild(new com.thinairapps.tag.wml.Text("It's a tie! "));
        }
        else
        { // the game is still in progress, let the user select the next move
            p.addChild(new com.thinairapps.tag.wml.Text("Enter the cell # for your next move ↙
                : "));
            int rowNum, colNum;
            String cellNumString;
            for (rowNum = 0; rowNum < theBoard.NUM_ROWS; rowNum++) {
                for (colNum = 0; colNum < theBoard.NUM_COLS; colNum++) {
                    if (theBoard.emptyAt(rowNum, colNum)) {
                        cellNumString = Integer.toString((rowNum * theBoard.NUM_COLS) +    ↙
                            colNum + 1);
                        p.addChild(new com.thinairapps.tag.wml.Anchor(path + "?row=" +     ↙
                            Integer.toString(rowNum) +
                            "&amp;col=" + Integer.toString(colNum) +                        -
                            "&amp;rnd=" + randNum + "&amp;sid=" +                           -
                                                                sessionID, "",
new com.thinairapps.tag.wml.Text(cellNumString)));                                           -
                    }
                }
            }
        }

        p.addChild(new com.thinairapps.tag.wml.Anchor(path + "?action=clear&amp;rnd=" +     ↙
            randNum +
                                                    "&amp;sid=" + sessionID, "",
                                                    new com.thinairapps.tag.wml.Text("Play ↙
                                                      a new game")));

        theCard.addChild(p);

        deck.addChild(theCard);

        resultString = deck.render();

        return resultString;
    }


    private void playTurn(String sessionID, TicTacToeBoard theBoard)
    {

        Character StartingPlayer, SecondPlayer;
```

```java
        //the cache for this session
        Hashtable cache = null;

        try
        {
            //get the cache for this session...
            cache = access.getSessionCache(sessionID);
        }
        catch (NoSuchSessionException e) { }

        Integer GameState = (Integer)cache.get("GameState");

        // did the user just enter a move?
        if (props.getProperty("row") != null)   {

            int rowNum = Integer.parseInt(props.getProperty("row"));
            int columnNum = Integer.parseInt(props.getProperty("col"));

            // is the move valid?
            if (theBoard.emptyAt(rowNum, columnNum)) {

                theBoard.placePiece('X', rowNum, columnNum);

                if (theBoard.playerWon('X')) {
                    GameState = USER_WON;
                } else if (theBoard.boardFull()) {
                    GameState = TIE_GAME;
                } else {

                    //GameState = (Integer)cache.get("GameState");
                    makeMove(theBoard);

                    if (theBoard.playerWon('O')) {
                        GameState = CONNECTOR_WON;
                    } else if (theBoard.boardFull()) {
                        GameState = TIE_GAME;
                    }
                }
            }
        }


        // if it's not a valid move, do nothing
        // if no move was entered, clear the board and start a new game
        else {
            GameState = IN_PROGRESS;

            theBoard.init();

            if (cache.get("SecondPlayer") != null) {
                StartingPlayer = (Character)cache.get("SecondPlayer");
            } else {
                StartingPlayer = new Character('X');
            }

            if (StartingPlayer.charValue() == 'X') {
                SecondPlayer = new Character('O');
            } else {
                SecondPlayer = new Character('X');
            }

            cache.put("StartingPlayer", StartingPlayer);
            cache.put("SecondPlayer", SecondPlayer);

            if (StartingPlayer.charValue() == 'O') {
                makeMove(theBoard);
            }
        }
        cache.put("GameState", GameState);
    }



    /**
     * This is a simple exception rendering method.
     *
     * @param message the message to be presented to the user
     * @return the rendered HTML page deck
     */
    private String renderException (String message)
    {
        //create the page
        HTMLTagDocument page = new HTMLTagDocument();

        Body body = new Body();

        //set the background color
        body.addAttribute("bgcolor","#ffffff");

        body.addChild(new com.thinairapps.tag.html.Text(message));
        body.addChild(new com.thinairapps.tag.html.Break());

        String resultString = body.render();

        return resultString;
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

import java.util.*;
import java.io.*;

class TicTacToeBoard extends Object
{

    final int NUM_ROWS = 3;
    final int NUM_COLS = 3;
    final char EMPTY_SPACE = '?';

    //The board is represented as an array of characters
    private char PieceAt[][] = new char[NUM_ROWS][NUM_COLS];

    //Set each cell in the board to be blank
    public void init()
    {
        int rowCount, colCount;

        for (rowCount = 0; rowCount < NUM_ROWS; rowCount++)
        {
            for (colCount = 0; colCount < NUM_COLS; colCount++)
            {
                PieceAt[rowCount][colCount] = EMPTY_SPACE;
            }
        }
    }



    /**
     * Returns the character occupying the cell at row number rowNum
     * and column number columnNum
     */
    public char pieceOccupying(int rowNum, int columnNum)
    {
        return PieceAt[rowNum][columnNum];
    }



    /**
     * Returns whether the cell at row number rowNum and column
     * number columnNum is empty
     */
    public boolean emptyAt(int rowNum, int columnNum)
    {
        return (PieceAt[rowNum][columnNum] == EMPTY_SPACE);
    }



    /**
     * Inserts a piece of type player at the cell with
     * row number rowNum and column number columnNum
     */
    public void placePiece(char player, int rowNum, int columnNum)
    {
        PieceAt[rowNum][columnNum] = player;
    }



    /**
```

```
     * Returns whether or not the board is completely
     * filled with pieces
     */
    public boolean boardFull()
    {
        int rowCount, colCount;

        for (rowCount = 0; rowCount < NUM_ROWS; rowCount++)
        {
            for (colCount = 0; colCount < NUM_COLS; colCount++)
            {
                if (PieceAt[rowCount][colCount] == EMPTY_SPACE)
                    return false;
            }
        }
        return true;
    }



    /**
     * Returns whether or not the player using the character
     * 'player' (either X or O) has a straight line of pieces in some
     * direction on the board
     */
    public boolean playerWon(char player)
    {
        int rowCount, colCount;
        boolean Won;

        //Check for horizontal win
        for (rowCount = 0; rowCount < NUM_ROWS; rowCount++)
        {

            Won = true;
            for (colCount = 0; colCount < NUM_COLS; colCount++)
            {
                if (PieceAt[rowCount][colCount] != player) Won = false;
            }
            if (Won) return true;
        }

        //Check for vertical win
        for (colCount = 0; colCount < NUM_COLS; colCount++)
        {

            Won = true;
            for (rowCount = 0; rowCount < NUM_ROWS; rowCount++)
            {
                if (PieceAt[rowCount][colCount] != player) Won = false;
            }
            if (Won) return true;
        }

        //Check for diagonal win
        Won = true;
        for (rowCount = 0, colCount = 0; rowCount < NUM_ROWS; rowCount++, colCount++)
        {
            if (PieceAt[rowCount][colCount] != player) Won = false;
        }
        if (Won) return true;

        Won = true;
        for (rowCount = 0, colCount = NUM_COLS - 1; rowCount < NUM_ROWS; rowCount++,
            colCount--)
        {
            if (PieceAt[rowCount][colCount] != player) Won = false;
        }
        if (Won) return true;
        return false;
    }
}
```

```
===============================================================================

                         WebScraper Sample Application

                         Wireless SDK for ThinAir Server

===============================================================================


-----------------
About This Sample
-----------------

This sample application contains both a Connector (SimpleWebConnector.java) and
a Provider (SimpleWebProvider.java). These two pieces work together to combine:

1. data access - via the ThinAir StoreProvider API

with

2. device specific rendering - via the Device detection facilities in ThinAir
   Server and the markup generating Tag Libraries

into one distributed solution.

When you contact the SimpleWebConnector with either a wireless device or web
browser you will receive a UI asking for a URL.  Enter a web address. The
Connector will contact the SimpleWebProvider to fetch the page.  The
SimpleWebProvider will retrieve the web page, eliminate all markup and
unprintable characters, and return the raw text to the Connector.  The
Connector will then render the page in approximately 1K chunks to each device
in its own markup.  You can scroll through the entire page, asking the
Connector for 'More' data via a link at the bottom of each screen.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * devices.jar


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - sample connector configuration file

    provider.ini - sample provider configuration file

    webScraper.html - a static HTML page that can be used by a Palm Pilot
    PQA

    WebScraper.jar - compiled Java code

    /src - Java source files for both the Connector and Provider


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  Be sure to
include the .jar files above in your CLASSPATH.

Install the Connector classes and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.
```

```
Install the Provider classes and provider.ini configuration file into a
subdirectory of ThinAir Server's /Providers subdirectory, given a name of
your choice.

The WebScraper root directory contains, for the sake of simplicity, a single
WebSCraper.jar file containing all the classes used by either the Connector or
the Provider (or both). Instead of dividing up the application into two sets of
classes, you can run both Connector and Provider by just placing a copy of this
file in both directories.

Start the ThinAir Server.  It should load SimpleWebConnector and
SimpleWebProvider and initialize both.


----------------
Using the Sample
----------------

Wait until the ThinAir Server has started and both the Connector and Provider
have been loaded and initialized.  From your wireless device, or web browser,
enter the IP address listed as the value for ApplicationPath in connector.ini
(your ThinAirServer IP address), followed by /samples/web.  For a machine with
IP address 111.222.12.34 this would be:

     http://111.222.12.34/samples/web

Follow the on-screen instructions.

Supported devices include WAP phones, HDML phones, Palm Pilots, Windows CE
devices, desktop web browsers, and GO America/GO RIM pagers.  To create a PQA
application for the Palm VII that integrates with the ThinAir Server, you
will need to understand and use "Web Clipping" technology from Palm.  Web
Clipping involves essentially creating HTML interfaces into your applications.
For your convienence, an HTML file (webScraper.html) has been provided for
this purpose.  To find out more about creating PQAs and Web Clipping
technology, visit: http://www.palmos.com/dev/tech/webclipping/


================================================================================

                          Last updated: 11.13.2000

                    Copyright 1999, 2000 ThinAirApps Inc.

================================================================================
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.device.*;

//Rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

//Core Java API
import java.util.*;



/**
 * @(#)WMLRenderer.java
 *
 * Utility class containing static methods for rendering output in wml
 */
public class WMLRenderer
{
    /**
     * Generate a WML page with a GUI with which the user can enter a URL
     *
     * @param connectorName the name of the Connector
     * @param path the HTTP path that maps to this Connector
     * @param reqProps the parameters of the original request + the session identifier
     */
    public static String showURLInputUI(String connectorName, String path, Properties
        reqProps) throws Exception
    {
        WMLTagDocument deck = new WMLTagDocument();

        SingleInputCard card = new SingleInputCard("url", "Enter URL");

        //Print the name of the Connector at the head of the card
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER, Paragraph.MODE_WRAP);
        p.addChild( new Text("Welcome to "+connectorName) );
        p.addChild( new Break() );
        card.addChild(p);

        // Construct the request URL
        // action (a) == get
        // url (url) == $url (to be filled in by the WML input element)
        // page number (pn) == start with the first chunk of data
        // session ID (sid) == determined in Connector and passed in request props param
        StringBuffer sb = new StringBuffer(56);
        sb.append( path );
        sb.append("?a=get&amp;url=$(url)&amp;pn=0&amp;sid=");
        sb.append( reqProps.getProperty("sid") );
        sb.append("&amp;rnd=");

        //Append a random number to combat caching
        sb.append( Math.random() );
        String url = sb.toString().trim();

        card.buildCard(url, "Enter URL: ", "url", "*" + Input.FORMAT_ANY_LCASE_CHANGEABLE );

        deck.addCard(card);

        return deck.render();
    }
    /**
     * Generate a page displaying a portion of the requested web page
     *
     * @param connectorName displayed at the top of the page
     * @param path used to issue another request
     * @param reqProps the properties of the original HTTP request
     * @param page the actual page text
     * @param more indicates whether any more data is available
     */
    public static String showURLOutput(String connectorName, String path, Properties reqProps↙
        ,
                                       String page, boolean more)
    {
        String sessionID = reqProps.getProperty("sid");
        String url = reqProps.getProperty("url");

        WMLTagDocument deck = new WMLTagDocument();

        DisplayCard card = new DisplayCard("page", connectorName);

        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER, Paragraph.MODE_NOWRAP);
        p.addChild( new Text(url) );
        card.addChild(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_WRAP);
        p.addChild( new Text(page) );
        p.addChild( new Break() );

        //If there is more text to display, add an anchor to request the next 1K
        StringBuffer sb;

        if (more)
        {
            int pn = Integer.parseInt( reqProps.getProperty("pn") );

            //Build the request url
            sb = new StringBuffer(56);
            sb.append(path);
            sb.append("?a=get&amp;url=");
            sb.append( url );
            sb.append("&amp;pn=");

            //Increment the page number
            sb.append( String.valueOf( ++pn ) );
            sb.append("&amp;sid=");
            sb.append( sessionID );
            sb.append("&amp;rnd=");

            //Append a random number to combat caching
            sb.append(Math.random());

            String href = sb.toString().trim();

            Anchor anchor = new Anchor( new Go(href, false), new Text("More") );
            p.addChild(anchor);
        }
        card.addChild(p);


        //Create a back button to take use back to the request page
        sb = new StringBuffer(56);
        sb.append(path);

        //Append an empty action (a) so that the Connector returns the input page
        sb.append("?a=&amp;sid=");
        sb.append(sessionID);
        sb.append("&amp;rnd=");
        //Append a random number to combat caching
        sb.append(Math.random());

        Do button = new Do(Do.TYPE_ACCEPT, new Go( sb.toString().trim(), false ));
        button.addAttribute("label", "Back");
        card.addChild(button);

        deck.addChild(card);

        String s = deck.render();
        return s;
    }


    /**
     * Generate a page describing an error that has occured
     *
     * @param e Exception which you wish to render
     */
    public static final String renderException(Exception e)
    {
      WMLTagDocument deck = new WMLTagDocument();
      DisplayCard card = new DisplayCard("Error", "Error");

      card.buildCard("Error: "+e.getMessage(), Paragraph.ALIGN_LEFT);

      deck.addCard(card);
      return deck.render();
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.provider.*;
import java.util.*;



/**
 * @(#)WebProviderResult.java
 *
 * This object is generated on the Provider to contain the results of a
 * web page query.  The Connector unwraps it and displays the returned text
 */
public class WebProviderResult extends StoreItem
{
    private String pageText;

    /**
     * Create a new WebProviderResult containing the processed text from a web page
     *
     * @param String text
     */
    public WebProviderResult(String text)
    {
        super();
        pageText = text;
    }


    /**
     * @return String the processed web page text
     */
    public String getText()
    {
        return pageText;
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core Java API
import java.net.*;
import java.util.*;
import java.io.*;



/**
 * @(#)urlTool.java
 *
 * This utility processes an HTML page and removes all markup
 */
public class urlTool
{
    private final static String SPECIAL_CHARS[] = { "&nbsp;",
                                                     "&amp;",
                                                     "&#174",
                                                     "&#38;",
                                                     "&#183",
                                                     "&#46;",
                                                     "&",
                                                     "$" };

    /**
     * Format a url to begin with http if it doesn't already then call process page
     */
    public static String getPageAndProcess(String page, int skipLines)
    {
        if (!page.toLowerCase().startsWith("http://"))
            page = "http://" + page;

        return processPage(getPage(page),skipLines);
    }



    /**
     * Remove all markup from a page and return it
     *
     * @param pageText - HTML contents
     * @param skipLines - return everything after this many lines
     */
    protected static String processPage (String pageText, int skipLines)
    {
        pageText = removeJS(pageText);
        pageText = removeTags(pageText);
        pageText = removeBlankLines(pageText);
        pageText = removeSpecial(pageText);

        return pageText;
    }



    /**
     * Format a url to begin with http if it doesn't already
     */
    protected static String checkURL(String page)
    {
        if (! page.toLowerCase().startsWith("http://"))
                page = "http://" + page;

        return page;
    }



    /**
     * Remove all blank lines and new lines
     *
     * @param text source String
     */
    protected static String removeBlankLines(String text)
    {
        StringBuffer output = new StringBuffer();

        StringTokenizer st = new StringTokenizer(text,"\n");

        String line;

        if (!st.hasMoreTokens())
        {
            output.append(text);
        }
        else
        {
            while(st.hasMoreTokens())
            {
                line = st.nextToken();
                line = line.trim();

                if (line.length() > 0)
                    output.append(line + "\n");
            }
        }
        return output.toString();
    }


    /**
     * Remove all script tags from HTML
     *
     * @param htmlCode source HTML
     */
    protected static String removeJS (String htmlCode)
    {
        htmlCode = removeTagPairContent(htmlCode, "<script", "</script>");
        htmlCode = removeTagPairContent(htmlCode, "<!-", "->");

        return htmlCode;
    }


    /**
     * Remove pairs of opening and closing tags
     */
    protected static String removeTagPairContent (String htmlCode, String start, String end)
    {
        int i = 0;
        int endIdx = 0;

        while((i = htmlCode.indexOf(start,i)) > 0)
        {
            endIdx = htmlCode.indexOf(end,i);
            if (endIdx == -1)
                break;
            htmlCode = htmlCode.substring(0,i)+htmlCode.substring(endIdx+end.length(),
                htmlCode.length());
            i = htmlCode.indexOf(start,i);
        }
        return htmlCode;
    }



    /**
     * Remove the text specified in SPECIAL_CHARS
     */
    protected static String removeSpecial(String htmlCode)
    {
        int numSpecial = SPECIAL_CHARS.length;
        int indexes[] = new int[numSpecial];

        int len = htmlCode.length();
        int newLen = len;

        char dummy = (char) 0;
        char buf[] = htmlCode.toCharArray();

        int i, j, k, index;

    outer:
        while (true)
        {

            // Count how many SPECIAL_CHARS have been found
            k = 0;

        inner:

            // Look through all the SPECIAL_CHARS
            for (i = numSpecial; --i >= 0; )
            {

                // No more SPECIAL to be found
                if (indexes[i] == -1)
                {

                    // Have ALL SPECIAL been found?
                    if (++k == numSpecial)
                        break outer;

                    //Skip and keep LOOKING
                    else
                        continue inner;
                }
                else
                {

                    //Look for more of SPECIAL
                    index = htmlCode.indexOf(SPECIAL_CHARS[i], indexes[i]);

                    //No more SPECIAL
                    if (index == -1)
                    {
                        //Mark as all done
                        indexes[i] = index;

                        //Continue to next
                        continue inner;
                    }
                }

                // Replace all chars in SPECIAL with dummy char
                for (j = SPECIAL_CHARS[i].length(); --j >= 0; )
                {
                    buf[ index + j ] = dummy;
                    --newLen;
                }

                //Advance indexes[i] to avoid repeats
                indexes[i] = index + 1;
            }
        }



        StringBuffer sb = new StringBuffer(newLen);

        //Copy all non-dummy chars into the return array
        for (i = 0; i < len; i++) {
            if (buf[i] == dummy)
                continue;
            else
                sb.append(buf[i]);
        }
        return sb.toString().trim();
    }




    /**
     * Remove anything that starts with a < and ends with a >
     *
     * @param htmlCode source HTML
     */
    protected static String removeTags (String htmlCode)
    {
        StringBuffer results = new StringBuffer();

        StringTokenizer st = new StringTokenizer(htmlCode,"<");

        String text = null;

        while(st.hasMoreTokens())
        {
            text = st.nextToken();
            text = text.substring(text.indexOf(">")+1);

            if (text.length() > 0 && !text.equals("\n") && !text.equals(" "))
                results.append(text + " ");
        }

        return results.toString();
    }


    /**
     * Fetch a page and return it
     */
    protected static String getPage (String urlString)
    {
        try
        {
            URLConnection uc = new URL(checkURL(urlString)).openConnection();
            return getStringFromStream(uc.getInputStream());
        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }
    }
    /**
     * Read a page from an input stream
     */
    protected static String getStringFromStream(InputStream is) throws Exception
    {
        StringBuffer sb = new StringBuffer(50000);

        BufferedReader br = new BufferedReader(new InputStreamReader(is));

        String line = br.readLine();

        while(line != null)
        {
            sb.append(line + "\n");
            line = br.readLine();
        }
        return sb.toString().trim();
    }


    /**
     * Skip this number of lines into the page
     */
    protected static String skipLines(String page, int lines)
    {
        if (lines == 0) return page;

        StringTokenizer st = new StringTokenizer(page,"\n");

        if (st.countTokens() > lines)
        {
            page = "";

            for (int i = 0; i < lines; i++)
                st.nextToken();

            while(st.hasMoreTokens())
                page = page + st.nextToken() + "\n";
        }

        return page;
    }
}
```

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.exception.*;

//Core Java API
import java.util.*;



/**
 * @(#)SimpleWebProviderContext
 *
 * Provides static information for and about the SimpleWebProvider
 *
 */
public class SimpleWebProviderContext extends StoreProviderContext
{
    public static final short WEB_PROVIDER_RESULT = 125;

    // Version information
    protected static final String VERSION      = "1.2";
    protected static String APP_NAME           = "WebScraper";
    protected static final String MANUF_NAME   = "ThinAirApps";
    protected static final String MANUF_CONT   = "www.ThinAirApps.com";
    protected static final String BUILD        = "1";
    protected static final Date   APP_RELEASED = new Date ();

    private Properties props;



    /**
     * Determines if the context has optional user-editable properties. Implementors should
     * return true if they can offer optional Properties to the user, but do not require   ↙
       these
     * properties to be set in order to correctly serve user connections.  StoreProviders may↙
       have
     * both property types.
     *
     * @return A boolean indicating whether or not the context has optional properties.
     */
    public boolean hasOptionalProps()
    {
        return false;
    }



    /**
     * @return ProviderObjectSet indicating the friendly and class names of StoreItem       ↙
        subclasses
     *          understood by this StoreProvider.
     */
    public StoreProviderType getType()
    {
        //Not used by this Provider
        return null;
    }



    /**
     * Called by a client to ask for product information on the Provider.
     *
     * @return StoreProviderInfo containing information on this Provider.
     */
    public StoreProviderInfo getInfo ()
    {
        return new StoreProviderInfo ( MANUF_NAME,
                                       MANUF_CONT,
                                       APP_NAME,
                                       VERSION,
                                       BUILD,
                                       APP_RELEASED );
    }


    /**
     * Tells the context to update its property set.  It will throw a                    ↙
       SPInvalidContextPropsException
     * if it does not accept the properties.
     *
     * @param props The new set of properties to commit.
     */
    public void updateProps(Properties p) { ; } // no provider-wide properties


    /**
     * @return a boolean indicating whether or not the context can offer required properties.
     */
    public boolean hasRequiredProps()
    {
        return true;
    }


    /**
     * Retrieves a ContextProperties object containing user-editable required and optional
     * properties
     *
     * @return ContextProperties object containing user-editable required and optional      ↙
       properties.
     */
    public ContextProperties getProps()
    {
        Properties required = (props == null) ? new Properties() : (Properties) props.clone ↙
            ();
        return new ContextProperties( new Properties(), required);
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.provider.*;

//Core Java API
import java.util.*;
import java.net.*;
import java.io.*;



/**
 * @(#)SimpleWebProvider.java
 *
 * This provider contacts an HTTP server, strips out all
 * of the markup, and returns its content to the Connector
 */
public class SimpleWebProvider implements StoreProvider
{
    private static int counter = 0;

    private SimpleWebProviderContext myContext;
    private int instanceNumber;
    private SupportedItems supportedItems;



    /**
     * Build a new SimpleWebProvider..
     */
    public SimpleWebProvider()
    {
        instanceNumber = ++counter;
        System.out.println("Starting SimpleWebProvider "+instanceNumber+"...");
    }



    /**
     * Create a connection to the back-end data store and retrieve SupportedItems.  Each      ↙
       instance
     * of a Provider represents a connection with the back-end store.  A Connector writer     ↙
       must
     * execute StoreProviderProxy's connectUser method before performing any other action     ↙
       with
     * regard to the Provider.
     *
     * The SupportedItems object that is returned by the connectUser method wraps a Vector of
     * SupportedItem objects, indicating what items and actions a StoreProvider supports for ↙
       a
     * given user.
     *
     * @return set of supported items based on user's access
     */
    public SupportedItems connectUser(StoreProviderLogin login, StoreProviderContext context)
    {
        // Because this Provider uses a stateless connection for each GET request,
        // there is no backend session to initialize here

        supportedItems = new SupportedItems();
        String name = SimpleWebProviderContext.APP_NAME;
        String location = null;
        try {
            location = InetAddress.getLocalHost().getHostAddress();
        } catch (Exception e) {
            location = "localhost";
        }

        // Support no actions
        short actions[] = new short[0];
        SupportedItem item = new SupportedItem(SimpleWebProviderContext.WEB_PROVIDER_RESULT, ↙
            name, location, actions);
        supportedItems.addItem(item);

        return supportedItems;
    }



    /**
     * The disconnectUser() method logs a user off of a Provider.  The Provider cannot be
     * used again until another user is logged on via the connectUser method.
     */
    public void disconnectUser()
    {
        // Similarly, there is nothing to do at disconnect time
        System.out.println("SimpleWebProvider "+instanceNumber+" shutting down.");
    }



    /**
     * Queries the StoreProvider for the locations it supports for the currently connected
     * user and returns the list.
     *
     * For this simple Provider there is no user data location
     *
     */
    public UserDataLocations getLocations(UserDataLocationRequest req)
    {
        return null;
    }



    /**
     * UserDataActions tell the provider to modify the backend data store in some way.
     * The only allowed modifications or "actions" are those specified when the user
     * logs on via connectUser.
     *
     * This simple provider does not support any actions
     *
     * @param action describes the requested action
     */
    public UserDataActionResponse doUserDataAction(UserDataAction action)
    {
        return null;
    }



    /**
     * The getUserData method is the means by a request is made for data from the data store.
     * This method is not used to performs action on data.
     *
     * This Provider uses UserDataRequests to retrieve information from some back-end HTTP  ↙
       server.
     * It processes the request by extracting the url from the request, contacts the web  ↙
       server,
     * retrieves the data, and returns it to the Connector.
     *
     * @param request represents the UserData request object
     */
    public UserData getUserData(UserDataRequest request)
    {
        ItemRequest itemReq = request.requests[0];

        // Prepare the return object
        UserData ud = new UserData();
        ud.responses = new ItemRequestResponse[ 1 ];
        ud.responses[0] = new ItemRequestResponse();
        ud.responses[0].request = itemReq;

        short type = itemReq.itemType;

        // Verify that the request is of the correct itemType
        if (type != SimpleWebProviderContext.WEB_PROVIDER_RESULT)
            throw new RuntimeException("Unknown item request type: "+type);

        // The requested URL is wrapped in a StringBound
        StringBound bound = (StringBound) ud.responses[0].request.bounds[0];
        String url = (String) bound.getValue();

        // Use the URL request tool to retrieve the page and process the results
        String data = urlTool.getPageAndProcess(url, 0);
        WebProviderResult result = new WebProviderResult(data);

        // Finish loading the return object
        ud.responses[0].items = new StoreItems();
        ud.responses[0].items.addElement(result);

        return ud;
    }
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;

//Core Java API
import java.net.*;
import java.io.*;
import java.util.*;



/**
 * @(#)SimpleWebConnector.java
 *
 * This Connector provides wireless clients with a simple user interface for requesting
 * a WWW page.  The web page is stripped down and returned as raw text.
 */
public class SimpleWebConnector implements Connector
{
    //Declare variables global to this Connector
    private ConnectorAccess connectorAccess;
    private String connectorName;
    private String path;

    //The maximum page length to be returned (in bytes)
    private static final int MAX_PAGE = 400;



    /**init() is called by the ThinAirServer when the Connector is loaded.  It provides the  ↙
        Connector
     * with resources it needs to interact with the ThinAirServer.
     *
     * @param name indicates the friendly name of this Connector application.  It is a String
     *             derived from connector.ini and this sample does utilize it.
     * @param path is the URL path to this Connector application.  It is a String derived
     *             from connector.ini and this sample does utilize it.
     * @param iniProps is a Properties object containing developer assigned,                  ↙
        connector-specific
     *             properties.  It is derived from connector.ini and this sample does not     ↙
        utilize it.
     * @param ca is the one-and-only interface a Connector obtains to gain access to the      ↙
        runtime services
     *             (such as session and user profile management) offered by the ThinAir       ↙
        Server to running
     *              Connectors.  This sample uses it.
     * @param appLog is used for logging.  This sample does not use it.
     */
    public void init(String name, String path, Properties iniProps, ConnectorAccess ca, com. ↙
        thinairapps.platform.connector.ApplicationLog appLog)
    {
        connectorName = name;
        connectorAccess = ca;
        this.path = path;
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a        ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing   ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used  ↙
       to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
       to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
       connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this       ↙
       Connector supports.
     */
    public String[] getDevices()
    {
        String[] devices = { "TA_WAP", "TA_UP_WAP", "TA_NOKIA_WAP",
                             "TA_PALM_VII", "TA_GOWEB_PALM", "TA_OMNISKY", "TA_HTML" };
        return devices;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming       ↙
       request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
       the server
     * receives a request from a type of device that the Connector indicates it supports,    ↙
       destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
       the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
       this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
       information
     * on the capabilities of the particular device making the request.
     *
     * @param reqProps - represents the HTTP request
     * @param device - the actual wireless device instance making the request
     * @param out - the OutputStream to write back the response
     */
    public void handle(Properties reqProps, Device device, OutputStream out) throws           ↙
        IOException
    {
        //This will hold the return markup
        String result = null;

        //Check the URL to see what action is being requested by the client
        String action = reqProps.getProperty("a");

        try
        {
            //Is the user returning or is it their first time using the Connector?
            String sessionID = reqProps.getProperty("sid");
            if (sessionID == null || (! connectorAccess.sessionValid(sessionID)))
            {
                //Create a session for this user
                //The session ID will be passed back and forth in the request URL
                sessionID = connectorAccess.createProviderSession(SimpleWebProviderContext.   ↙
                    APP_NAME);
                reqProps.put("sid", sessionID);
                initProvider(sessionID);
            }

            //If this is the first request by the device
            if (action == null || action.equals(""))
            {
                //Return the URL input page
                result = showURLInputUI(reqProps, device);
            }
            else if (action.equals("get"))
            {
                //If the user has requested the URL input page
                String pn = reqProps.getProperty("pn");

                if (pn.equals("-1"))
                {
                    result = showURLInputUI(reqProps, device);
                }

                //Use the backend Provider to retrieve a page and process it
                //Cache the entire page and dole it out in 1K chunks
                else
                result = getURL(reqProps, sessionID, device);
            }
            else
                throw new Exception("Unknown action: "+action);
        }
        catch (Exception e)
        {
            //Catch all exceptions and generate an error page
            e.printStackTrace();

            result = renderException(e, device);
        }
        out.write( result.getBytes() );
    }


    /**
     * Generate a page with a URL input user interface
     */
    protected String showURLInputUI(Properties reqProps, Device device) throws Exception
    {
        //Use different utility classes to render output depending on the markup required by ↙
            the device
        if (device instanceof WAPDevice)
            return WMLRenderer.showURLInputUI(connectorName, path, reqProps);
        else if (device instanceof HTTPDevice || device instanceof PalmVIIDevice || device   ↙
            instanceof GoWebPalmDevice ||
                 device instanceof OmniSkyDevice || device instanceof HTMLDevice )
                return HTMLRenderer.showURLInputUI(connectorName, path, reqProps);

        //This simple web Connector does not support all devices
        else
            throw new Exception("Device "+device.getProfile().getName()+ " not supported");
    }



    /**
     * Retrieve a page either from cache or from the backend Provider and return it in 1K     ↙
        chunks
     */
    public String getURL(Properties reqProps, String sessionID, Device device) throws         ↙
        Exception
    {
        Hashtable sessionCache = connectorAccess.getSessionCache( sessionID );

        // the requested URL
        String url = reqProps.getProperty("url");

        String page;
        String retString = null;
        int endIndex, length;
        //Is the user asking for MORE of a page that has already been retrieved,
        //or is this a request for a fresh page?
        String pn = reqProps.getProperty("pn");
        if (pn == null)
        {
            pn = "0";
            reqProps.put("pn", "0");
        }

        // If this is a first-time request
        if (pn.equals("0"))
        {
            // retrieve the data
            page = requestPage(sessionID, url);

            // store entire page in session cache
            sessionCache.put("page", page);

            // return first 1K of text
            length = page.length();
            if (length < MAX_PAGE)
            {
                endIndex = length;
                retString = page;
            }
            else
            {
                endIndex = MAX_PAGE;
                retString = page.substring(0, endIndex) + "...";
            }
        }
        else
        {
            //Retrieve the page from the session cache
            //Look at the pageNumber (pn)
            //Retrieve substring with chars (pn * MAX_PAGE) ---> (pn + 1) * MAX_PAGE
            //from the page
            int num = Integer.parseInt(pn);
            page = (String) sessionCache.get("page");
            length = page.length();
            endIndex = (num + 1) * MAX_PAGE;
            if (endIndex >= length)
                retString = "..." + page.substring(num * MAX_PAGE);
            else
                retString = "..." + page.substring( num * MAX_PAGE, endIndex );
        }

        //Use different utility classes to render output depending on the markup required by ↙
            the device
        if (device instanceof WAPDevice)
            return WMLRenderer.showURLOutput(connectorName, path, reqProps, retString          ↙
                , (endIndex < length) );

        else if (device instanceof HTTPDevice || device instanceof PalmVIIDevice || device    ↙
            instanceof GoWebPalmDevice ||
                 device instanceof OmniSkyDevice || device instanceof HTMLDevice )
            return HTMLRenderer.showURLOutput(device, connectorName, path, reqProps,           ↙
                retString, (endIndex < length) );

        // this simple web Connector does not support all devices - return error
        else
            throw new Exception("Device "+device.getProfile().getName()+ " not supported");
    }


    /*
     * Generate an exception page
     */
    private String renderException(Exception e, Device device)
    {
        //Use different utility classes to render output depending on the markup required by ↙
            the device
        if (device instanceof WAPDevice)
            return WMLRenderer.renderException(e);
        else if (device instanceof HTMLDevice)
            return HTMLRenderer.renderException(e);
        else
            return "ERROR: "+ e.getMessage();
    }



    /*
     * Initialize the Provider that we will contact again when the user enters a URL
     */
    private void initProvider(String sessionID) throws Exception
    {
        //Retrieve the StoreProviderProxy for an existing session
        StoreProviderProxy spProxy = connectorAccess.getStoreProvider(sessionID);

        //Construct this object for pedagogical purposes only
        StoreProviderLogin login = new StoreProviderLogin(null, null, null);
        SupportedItems supports = spProxy.connectUser(login);

        if (supports == null)
            throw new Exception("Error in connectUser.  Provider is unavailable");
    }



    /*
     *Request a page from the Provider and return it
     */
    private String requestPage(String sessionID, String url) throws Exception
    {
        //Retrieve the StoreProviderProxy for an existing session
        StoreProviderProxy spProxy = connectorAccess.getStoreProvider(sessionID);

        //Construct the request object
        UserDataRequest request = new UserDataRequest();
        request.requests = new ItemRequest[1];
        request.requests[0] = new ItemRequest();
        request.requests[0].itemType = SimpleWebProviderContext.WEB_PROVIDER_RESULT;
        request.requests[0].bounds = new Bound[1];
        request.requests[0].bounds[0] = new SimpleURLBound(url);

        //Contact the Provider and make the UserDataRequest
        UserData response = spProxy.getUserData(request);

        //Extract the data
        StoreItems items = response.responses[0].items;
        WebProviderResult result = (WebProviderResult) items.elementAt(0);

        return result.getText();
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.provider.*;



/**
 * @(#)SimpleURLBound.java
 *
 * Connectors and Providers can use this bound to request a specific URL
 *
 */
public class SimpleURLBound extends StringBound
{
    /**
     * Create a new SimpleURLBound to request a specific URL
     *
     * @param String the url to request
     */
    public SimpleURLBound(String url)
    {
        super(url, StringBound.COND_EQUALS);
    }
}
```

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.device.*;

//Rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;

//Core Java API
import java.util.*;



/**
 * @(#)HTMLRenderer.java
 *
 * A utility class containing static methods for rendering output in html
 */
public class HTMLRenderer
{
    /**
     * Generate a WML page with a GUI with which the user can enter a URL
     *
     * @param connectorName the name of the connector
     * @param path the HTTP path that maps to this Connector
     * @param reqProps the parameters of the original request + the session identifier
     */
    public static String showURLInputUI(String connectorName, String path, Properties
        reqProps) throws Exception
    {
        //Create the basic "page" object, used for all HTML document rendering
        HTMLTagDocument doc = new HTMLTagDocument();
        Head head = new Head();
        Meta meta = new Meta("name", "PalmComputingPlatform", "true");
        head.addChild(meta);
        head.addChild(new Title("Enter URL"));

        doc.setHead(head);

        //create the body
        Body body = new Body();
        Font font = new Font("geneva,arial", 3);
        Break br = new Break();

        //All children tags of this tag will be centered on the page
        Center center = new Center();
        center.addChild( new Text("<B>"+connectorName+"</B>") );
        center.addChild( br );
        center.addChild( new HorizontalRule() );
        center.addChild( br );

        font.addChild(center);

        //Create the form
        Form form = new Form("getURL", path, "GET");

        //Add form input elements
        form.addChild( new Input("hidden", "a", "get") );
        form.addChild( new Input("hidden", "pn", "0") );
        form.addChild( new Input("hidden", "sid", reqProps.getProperty("sid")) );
        form.addChild( new Text("Enter URL:&nbsp;") );
        TextField text = new TextField("url", "", 17);
        text.addAttribute( new Attribute("maxlength", "50") );
        form.addFormElement(text);
        form.addFormElement(new SubmitButton("Go"));

        font.addChild(form);

        body.addChild(font);
        doc.setBody(body);

        //Returns the entire rendered document text, suitable for display in an HTML browser
        return doc.render();
    }



    /**
     * Generate a page displaying a portion of the requested web page
     *
     * @param connectorName displayed at the top of the page
     * @param path used to issue another request
     * @param reqProps the properties of the original HTTP request
     * @param page the actual page text
     * @param more is there any more data available
     */
    public static String showURLOutput(Device device,String connectorName, String path,    ↙
        Properties reqProps,
                                        String page, boolean more)
    {
        //Retrieve the SessionID from the URL and create a reference to it
        String sessionID = reqProps.getProperty("sid");

        //Create the basic "page" object, used for all HTML document rendering
        HTMLTagDocument doc = new HTMLTagDocument();
        Head head = new Head();
        Meta meta = new Meta("name", "PalmComputingPlatform", "true");
        head.addChild(meta);
        head.addChild(new Title(connectorName));
        doc.setHead(head);

        //create the body
        Body body = new Body();

        //A tag indicated text styling parameters
        Font font = new Font("geneva,arial", 3);
        Break br = new Break();

        //Get the URL that the user entered
        String url = reqProps.getProperty("url");

        font.addChild( new Text(url) );
        font.addChild( br );
        font.addChild( new HorizontalRule() );
        font.addChild( br );
        font.addChild( new Text(page) );
        font.addChild( br );

        StringBuffer sb;

        //Turn simple links into anchor buttons
        Attribute button = new Attribute("BUTTON", "BUTTON") {
            public String render() { return "BUTTON"; }
        };

        //Get the page number that the user is on
        int pn = Integer.parseInt( reqProps.getProperty("pn") );

        //Get the request action
        String action = reqProps.getProperty("a");

        //If there's more of the page then construct the more link and increment the page    ↙
```

```
            number variable
        if (more)
        {
            // Build the request url
            sb = new StringBuffer(56);
            sb.append(path);
            sb.append("?a=get&amp;url=");
            sb.append( url );
            sb.append("&amp;pn=");

            // Increment the page number
            sb.append( String.valueOf( ++pn ) );
            sb.append("&amp;sid=");
            sb.append( sessionID );
            sb.append("&amp;rnd=");

            // Append a random number to combat caching
            sb.append(Math.random());

            String href = sb.toString().trim();

            Anchor anchor = new Anchor( "More", href, new Text("More"));
            anchor.addAttribute(button);

            font.addChild(anchor);
            font.addChild(new Text("&nbsp;&nbsp;&nbsp;"));
        }

        // Now for the back link...
        // Depending on the device version, make the back link point to the appropriate ↙
            request page
        if (device instanceof HTMLDevice)
        {
                //Build the request url
                sb = new StringBuffer(56);
                sb.append(path);
                sb.append("?a=get&amp;url=");
                sb.append( url );
                sb.append("&amp;pn=");

                // If there's more of the page, decrement the page number by 2 since we just ↙
                    incremented
                // it for the purposes of creating the 'more' link
                if (more)
                {
                    sb.append( String.valueOf( pn - 2 ) );
                }
                else
                {
                    sb.append( String.valueOf( --pn ) );
                }

                sb.append("&amp;sid=");
                sb.append( sessionID );
                sb.append("&amp;rnd=");

                //Append a random number to combat caching
                sb.append(Math.random());

                String href = sb.toString().trim();

                Anchor back = new Anchor( "Back", href, new Text("Back") );
                back.addAttribute(button);
                font.addChild(back);

                body.addChild(font);

                doc.setBody(body);
                return doc.render();
        }



      /**
       * If its a palm device...
       * To create a PQA application for the Palm VII that integrates with the ThinAir Server↙
        , you
       * will need to understand and use "Web Clipping" technology from Palm. Web Clipping   ↙
        involves
       * essentially creating HTML interfaces into your applications.  To find out more about
       * creating PQAs and Web Clipping technology, visit: http://www.palmos.com/dev/tech/   ↙
        webclipping/
       */
        else
        {
            Anchor back = new Anchor( "Back", "file:webScraper.pqa", new Text("Back") );
            back.addAttribute(button);
            font.addChild(back);

            body.addChild(font);

            doc.setBody(body);
            return doc.render();
        }
    }



    /**
     * Generate a page describing an error that has occured
     *
     * @param e - An exception which you wish to render
     */
    public static final String renderException(Exception e)
    {
        HTMLTagDocument doc = new HTMLTagDocument();
        Head head = new Head();
        Meta meta = new Meta("name", "PalmComputingPlatform", "true");
        head.addChild(meta);
        head.addChild(new Title("Enter URL"));
        doc.setHead(head);

        Body body = new Body();
        Break br = new Break();

        Font font = new Font("geneva,arial", 3);
        font.addChild( br );
        font.addChild(new Text("ERROR: "+e.getMessage()));
        body.addChild( font );

        doc.setBody( body );

        return doc.render();
    }
}
```

```
==========================================================================

                ThinAir Distributed File Store Provider
           Microsoft Windows NT/2000 Distribution, Version 1.1

                              README.TXT

==========================================================================


---------------------------
What's in this Distribution
---------------------------

The Distributed File Store Provider allows individual users with a "ThinAir Powered Enterprise" to
wireless enable access to a set of disk based documents. The Provider maps your file system directories
and file names to Groupware "Messages" and "Folders". The type of files supported by this release
include:
        -MS Word 2000 Documents (.doc)
        -HTML Pages (.html,.htm)
        -Plain Text (.txt)
        -XML Documents (.xml)
        -Java Source Code (.java)

This software also demonstrates the highly distributable capabilities of the ThinAir Server Platform.

--------------------------
Security
--------------------------

All authentication is done against existing NT Domain accounts. When you setup your TA Groupware Account,
you must supply the following information:

        Provider: your personal provider name (specified in provider.ini)
        Host: your NT Domain (i.e. "taaps_lan")
        UserName: your NT Account Login
        Password: your NT Account Password

Also, access to the file system is controlled by the same permissions granted to the user account which
is running the provider.

--------------------------
Installing the Application
--------------------------

Here's a step by step:

1) Copy the file "TAAuthUtils.dll" into your Windows or WinNT directory.

2) Open the file "provider.ini" edit the properties:

        -ProviderName=f.beano

        The ProviderName property must be a unique name. The scheme used above should be
        employed to help guarantee uniqueness within an NT domain. Replace "beano" with
        your NT login.

        -DefaultBasePath=\\tapdc\data\

        The DefaultBasePath is the "root" location which the File Store Provider should browse
        from.

        -UserDirectory:beano=\\tapdc\usr\beano\

        To specify a user specific directory, you can add "UserDirectory:<NT Login>" parameters
for every user. again, We recommend simply replacing "beano" with your NT Login.


3) Launch the "StartProvider.bat" file. This should register your provider with the TAS server running on
10.1.1.47 (our primary thinair server).

4) Now you need to setup an account with the TAS server in the same way you would for access to any
standard Groupware Provider.

        -Select your provider name from the drop-down list
        -For the host, enter our NT Domain "taaps_lan"
        -Enter your NT Login and password (or leave the password blank)
        -You can leave display, email blank
        -Name the account
        -Save it!

5) Now login! If your BaseDirectoryPath or UserDirectory points to a directory with only folders inside
of it, you first "inbox" view will be blank. If you browse folders, you should be able to see all of the
subdirectories. Select one with files, and your "inbox" should get populated by them.

--------------------------
Known Issues
--------------------------
-Some MSWord documents cannot be opened
-This has not been tested on Windows 98, though it may work

--------------------------
Support and Bug Reporting
--------------------------
Well, this is basically unsupported, but I would like to hear about certain bugs
you may find, specifically regarding problems reading certain types of documents.

================================================================================

                              Last updated: 8.18.2000

                       Copyright 1999, 2000 ThinAirApps, Inc.

================================================================================
```

```java
package thinairapps.groupware.storeproviders.file;

import java.io.*;

public class DocumentConverter
{

    public final static String TYPE_TEXT = "text/plain";
    public final static String TYPE_HTML = "text/html";
    public final static String TYPE_WORD = "application/msword";

    public static String readDocument (File file, String mimeType)
    {
        try
        {

        if (mimeType == null)
            return null;
        else if (mimeType.equals (TYPE_TEXT))
            return getASCII (file);
        else if (mimeType.equals (TYPE_HTML))
            return getHTML (file);
        else if (mimeType.equals (TYPE_WORD))
            return getMSWord (file);
        else
            return null;

        }
        catch (Exception e)
        {
            return null;
        }
    }

    public static String getASCII (File file)
    {
        String data = readFile (file);
        return data;
    }

    public static String getHTML (File file)
    {
        String data = readFile (file);
        return HTMLRipper.processPage (data);
    }

    public static String getMSWord (File file)
    {
        String data = readFile (file);

        int startIdx = data.indexOf("Ù");

        if (startIdx != -1)
            return data.substring (startIdx+1).trim();
        else
            return data;

    }

    private static String readFile (File file)
    {
        try
        {
            FileReader fis = new FileReader (file);
            BufferedReader reader = new BufferedReader (fis);
            StringBuffer out = new StringBuffer();
            String line = reader.readLine();

            while (line != null)
            {
                if (line.length() > 0)
                    out.append (line + "\n");

                line = reader.readLine();
            }

            return out.toString();
        }
        catch (IOException e)
        {
            return "Unable to read file: " + e;
        }
    }

}
```

```
package thinairapps.groupware.storeproviders.file;

import javax.mail.*;
import javax.mail.internet.*;
import javax.mail.event.*;
import java.util.*;

import javax.activation.*;


public class DocumentSender
{

    public static void sendDocument (String file, String server, String from, String to,
        String subject, String msgText1)
    {
        // create some properties and get the default Session
        Properties props = System.getProperties();
        props.put("mail.smtp.host", server);

        Session session = Session.getDefaultInstance(props, null);
        session.setDebug(false);

        try {
            // create a message
            MimeMessage msg = new MimeMessage(session);
            msg.setFrom(new InternetAddress(from));
            InternetAddress[] address = {new InternetAddress(to)};
            msg.setRecipients(Message.RecipientType.TO, address);
            msg.setSubject(subject);

            // create and fill the first message part
            MimeBodyPart mbp1 = new MimeBodyPart();
            mbp1.setText(msgText1);

            // create the Multipart and its parts to it
            Multipart mp = new MimeMultipart();
            mp.addBodyPart(mbp1);

            // create the second message part
            MimeBodyPart mbp2 = new MimeBodyPart();

                // attach the file to the message
            FileDataSource fds = new FileDataSource(file);
            mbp2.setDataHandler(new DataHandler(fds));
            mbp2.setFileName(fds.getName());

            mp.addBodyPart(mbp2);

            // add the Multipart to the message
            msg.setContent(mp);

            // set the Date: header
            msg.setSentDate(new Date());

            // send the message
            Transport.send(msg);

        } catch (MessagingException mex) {
            com.thinair.utils.DbgLog.logError (mex);
            Exception ex = null;
            if ((ex = mex.getNextException()) != null) {
                com.thinair.utils.DbgLog.logError (ex);
            }
        }
    }
}
```

```
package thinairapps.groupware.storeproviders.file;

import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.exception.*;


import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;

import com.thinair.utils.*;

import java.util.Vector;
import java.util.Enumeration;

import java.util.Properties;
import java.util.Date;
import java.util.StringTokenizer;

import java.io.*;

import com.ms.com.Variant;
import com.ms.wfc.app.Time;
import com.thinair.utils.win32.COMTypes;
import com.thinair.utils.DbgLog;
import com.ms.win32.Advapi32;

import com.ms.security.permissions.*;
import com.ms.security.*;
import com.ms.util.*;
/**
 * An File StoreProvider that offers the following types of objects from the
 * GroupwareObjectSet:
 *
 *      "message" - Message objects from a user's folder.  The item locations correspond to
    IMAP
 *                      folders, the default being INBOX.
 *
 * @author meyerwil
 */
public final class FileStoreProvider implements StoreProvider
{

    // Version info
    protected static final String VERSION      = "1.0 Early Access 2";
    protected static final int BUILD           = 1;
    protected static final String APP_NAME     = "ThinAir File Provider";
    protected static final String MANUF_NAME   = "ThinAirApps";
    protected static final String MANUF_CONT   = "www.ThinAirApps.com";
    protected static final Date   APP_RELEASED = new Date (100, 0, 5);

    private final static int DOC_LENGTH_MAX = 25000;

    public static final int FOLDER_SEPERATOR = 0x2F;

    static final Variant  V_FALSE = new Variant (false);
    static final Variant  V_TRUE = new Variant (true);
    static final Variant  V_NOPARAM;

    private static final int VB_DOUBLE = 5;
    private static final int VB_STRING = 8;

    public  static final Variant  V_PROPTAG_CONTAINERTYPE = new Variant (0x3613001EL);
    private static final Variant  V_PROPTAG_PUBLICSTORE_ROOT = new Variant (0x66310102L);
    private static final Variant  V_PROFILENAME = new Variant ("");
    private static final Variant  V_PW = new Variant ("");
    private static final Variant  V_PARENTHWND = new Variant ((int)-1);

    private int m_currLocationDepth;
    private static final String DEFAULT_LOC = "Files";

    private static String SMTP_HOST = "localhost";
    private static String MIME_TYPE_FILE_PATH = "./mime.types";

    private StoreProviderLogin login;

    private final static String NO_BODY_MSG = "Document type not supported for viewing.";

    static
        {
        V_NOPARAM = new Variant ();
        V_NOPARAM.noParam ();
        }

    private static Properties mimeTypeMap;

    /**
     * Constructs an IMAPStoreProvider instance with an always-used FetchProfile.
     *
     * @author meyerwil
     */
    public FileStoreProvider ()
    {
        if (mimeTypeMap == null)
        {
            try
            {
                loadMimeTypeFile (MIME_TYPE_FILE_PATH);
            }
            catch (Exception e)
            {
                e.printStackTrace ();
            }
        }
    }

    private String getBasePath (String login)
    {
        DbgLog.logStatus ("FileProv: looking up user basepath");

        return FileStoreProviderContext.getUserDirectory (login);

    }

    /**
     * Instances can share the javamail session, an LDAP handler, and the SMTP server through↙
        the
     * context object.  This method attempts to reuse shared resources or creates new ones if
     * needed.  Upon completion, the session, smtp and ldap members will be adjusted with any↙
        shared
     * resources.
     *
     * @param popContext The POPStoreProviderContext instance shared across POPStoreProviders
     *
     * @author meyerwil
     */
    private void setupSharedStuff (FileStoreProviderContext fileContext)
    {
        SMTP_HOST = fileContext.m_props.getProperty ("SMTPHost");
        MIME_TYPE_FILE_PATH = fileContext.m_props.getProperty ("MimeTypeFilePath");

    }

    private void authenticateUser (StoreProviderLogin login) throws                          ↙
        AuthenticationFailedException
    {
        //need to implement directory lookup here
        this.login = login;
        if (AuthenticateUserByUsernamePassword (login.name, login.password, login.host) != 0)
            throw new AuthenticationFailedException();


    }

    /** @dll.import("taauthutils.dll") */
    private static native int AuthenticateUserByUsernamePassword (String userName, String ↙
        password, String domain);


    /**
     * Logs a user on to an IMAP store using the information in login.  A Caller must log on ↙
        by
     * calling this function before doing anything else with the object.
     *
     * @param login   An object containing the required login information.
     * @param context A StoreProviderContext used for all StoreProvider instances.
     *
     * @return A SupportedItems object containing information on the supported actions and ↙
        item types
     *         for this user.
     *
     * @author meyerwil
     */
    public SupportedItems connectUser (StoreProviderLogin login, StoreProviderContext ↙
        context) throws ProviderException
    {
        short                    actions[];
        SupportedItems           supportedItems;
        SupportedItem            supportedItem;
        FileStoreProviderContext fileContext;

        // Make sure the password is correct
        DbgLog.logStatus ("FileGroupwareProvider.connectUser: Authenticating user");
        authenticateUser (login);

        DbgLog.logStatus ("FileStoreProvider.connectUser: Connecting...");
        if ((context == null) || !(context instanceof FileStoreProviderContext))
            throw new InvalidContextException ();
        fileContext = (FileStoreProviderContext)context;
        synchronized (this)
            {
            try
                {

                // Do item support stuff
                supportedItems = new SupportedItems ();
                actions = new short[2];
                actions[0] = Actions.ADD_NEW_GROUPWARE_ITEM;
                actions[1] = Actions.DELETE_GROUPWARE_ITEM;
                supportedItem = new SupportedItem (ItemTypes.MESSAGE, "Files", DEFAULT_LOC, ↙
                    actions);
                supportedItems.addItem (supportedItem);

                return supportedItems;
                }
            catch (Exception e4)
                {
                throw new ProviderException (e4 + ": " + e4.toString());
                }
            }
    }

    /**
     * Returns all of the folders provided by the IMAP host.  Location names will include
     * forward slashes ('/') to indicate the hierarchy.  This provider supports only Message ↙
        items
     * and the caller must therefore not request locations for other types of items.
     *
     * @param req The request forlocations meeting certain criteria.
     *
     * @return A UserDataLocations object containing the retrieved locations that match the
     *         requested criteria.
     *
     * @author meyerwil
     */
    public UserDataLocations getLocations (UserDataLocationRequest req) throws
        ProviderException
    {

        File dir = null;

        String rootPath = null;

        if (req.rootLocation != null && !(req.rootLocation.equals(DEFAULT_LOC)))
        {
            String path = getBasePath (login.name) + req.rootLocation.replace((char)
                FOLDER_SEPERATOR,'\\');
            dir = new File (path);
            rootPath = req.rootLocation;
        }
        else
        {
            dir = new File (getBasePath(login.name));
        }

        String[] dirList = dir.list ();
        String[] folderNames = new String[0];

        if (dirList != null)
        {
            Vector folders = new Vector();

            File file;

            for (int i = 0; i < dirList.length; i++)
            {
                file = new File (dir.getAbsolutePath() + File.separator + dirList[i]);
                if (file.isDirectory())
                {
                    if (rootPath != null)
                        folders.addElement(rootPath + ((char)FOLDER_SEPERATOR) + dirList[i]);
                    else
                        folders.addElement(dirList[i]);

                }
            }

            folderNames = new String [folders.size()];

            for (int i = 0; i < folders.size(); i++)
                folderNames[i] = (String)folders.elementAt(i);

        }

        return new UserDataLocations (req.itemType, folderNames);
    }

    /**
     * Closes a connection to the provider -- logs a user off.  This method MUST be called if
       a
     * user successfully logged-on using connectUser ().
     *
     * @author meyerwil
     */
    public void disconnectUser () throws ProviderException
    {

    }


    /**
     * Retreives files that match the incoming request.  This is the entry-point for
     * requesting any Message object  The allowable bounds/fields are as follows:
     *
     *          Message:
     *              (StringItemBound)
     *                  "subject"
     *                  "body"
     *                  "from"
     *              (DateItemBound)
     *                  "received"
     * @param req The request for specific data from the user's store.
     *
     * @return A UserData object containing the requested data.
     *
     * @author meyerwil
     */
    public UserData getUserData (UserDataRequest req) throws ProviderException
    {
        UserData data = new UserData();

        data.responses = new ItemRequestResponse[1];

        data.responses[0] = new ItemRequestResponse();
        data.responses[0].request = req.requests[0];

        data.responses[0].items = new StoreItems();
        ((StoreItems)data.responses[0].items).setIsLastAvailable (true);

        String path = getBasePath(login.name);

        if (req.requests[0].itemLocation != null && !(req.requests[0].itemLocation.equals     ↙
            (DEFAULT_LOC)))
            path += req.requests[0].itemLocation.replace('/',File.separatorChar);

        DbgLog.logStatus ("FileProv: accessing path: " + path);

        File dir = new File (path);

        String[] dirList = dir.list ();

        if (dirList != null)
            DbgLog.logStatus ("FileProv: got directory listing; size=" + dirList.length);
        else
            throw new AuthenticationFailedException();


        File file;
        Message msg;
        String mimeType;
        String fileExt;

        int startIdx = 0;

        if (req.requests[0].startID != null)
            startIdx = new Integer (req.requests[0].startID.substring(0,req.requests[0].     ↙
                startID.indexOf(":"))).intValue() + 1;

        for (int i = startIdx; i < dirList.length; i++)
        {
            file = new File (path + "\\" + dirList[i]);

            if (!file.isDirectory())
            {
                DbgLog.logStatus ("FileProv: examining file: " + file.getName());

                msg = new Message ();

                ((StoreItem)msg).setID (i + ":" + file.getAbsolutePath());

                msg.setReceivedDate (new Date(file.lastModified()));
                msg.setRead(false);
                msg.addRecipient (Message.RecipientType.TO,new GroupwareUserSMTPAddress(login↙
                    .name,login.name));

                mimeType = null;
                fileExt = "NONE";

                if (file.getName().indexOf (".") != -1)
                {
                    fileExt = file.getName().substring (file.getName().lastIndexOf (".")+1);
                    mimeType = mimeTypeMap.getProperty (fileExt.toLowerCase());
                }


                msg.setFrom (new GroupwareUserSMTPAddress(fileExt.toUpperCase(),fileExt.     ↙
                    toUpperCase()));

                String body = null;

                try
                {
                    body = DocumentConverter.readDocument (file, mimeType);
                }
                catch (Exception e)
                {
                    DbgLog.logError ("error reading file: " + e);
                    DbgLog.logError (e);
                }

                if (body != null)
                {
                    if (body.length() > DOC_LENGTH_MAX)
                        body = body.substring (0, DOC_LENGTH_MAX);

                    msg.setSubject ("*" + file.getName());
                    msg.setBody (body);
                }
                else
                {
                    msg.setSubject (file.getName());
                    msg.setBody (NO_BODY_MSG);
                }

                data.responses[0].items.addElement (msg);
            }


            if (data.responses[0].items.size() == req.requests[0].max)
            {
                if (i < dirList.length-1)
                    ((StoreItems)data.responses[0].items).setIsLastAvailable (false);
                else
                    ((StoreItems)data.responses[0].items).setIsLastAvailable (true);

                break;
            }
        }

        DbgLog.logStatus ("FileProv: returning message set");

        return data;
```

```
}
/**
 * Performs some action on the user's data.  This provider only supports
    "add", "delete", and
 * "markseen" actions, passed as UserDataAdd, UserDataDelete, and UserDataMarkSeen
    objects.
 * Adding an item means sending an email message.
 *
 * @param action The action to be performed on the user data.
 *
 * @return A UserDataActionResponse containg the results of the action.
 *
 * @author meyerwil
 */
public UserDataActionResponse doUserDataAction (UserDataAction action) throws
    ProviderException
{

    thinairapps.groupware.api.Message  gwMsg = null;

    synchronized (this)
        {
        if (action instanceof AddNewGroupwareItem)
            {
            GroupwareUserAddress recips[];
            GroupwareUserAddress tempAddr;

            // We're going to "add" the message, meaning send it
            gwMsg = (thinairapps.groupware.api.Message)((AddNewGroupwareItem)action).
                getItem ();
            try
                {

                 if (gwMsg instanceof ForwardedMessage)
                    {

                        String filePath = ((ForwardedMessage)gwMsg).getOriginalID ();
                        int sIdx = filePath.indexOf (":") + 1;
                        filePath = filePath.substring (sIdx);
                        File file = new File (filePath);
                        String from = gwMsg.getFrom ().getAddress();

                        StringBuffer to = new StringBuffer ();
                        GroupwareUserAddress[] toz = gwMsg.getRecipients (Message.
                            RecipientType.TO);
                        for (int n = 0; n < toz.length; n++)
                        {
                            to.append (toz[n].getAddress ());

                            if (n+1 < toz.length)
                                to.append (";");
                        }

                        String subject = gwMsg.getSubject ();
                        String message = gwMsg.getBody ();

                        DbgLog.logStatus ("Sending file to '" + to.toString() + "': " +
                            file.getName() + " via " + SMTP_HOST);
                        DocumentSender.sendDocument (file.getAbsolutePath (), SMTP_HOST,
                            from, to.toString(), subject, message);

                    }
                  else
                      throw new InvalidActionException ();
                }
            catch (Exception e)
                {
                if (e instanceof ProviderException)
                    throw (ProviderException)e;
                DbgLog.logError ("Caught unknown exception adding item: " + e.toString
                    ());
                throw new ProviderException ("Unable to create item.");
                }
            }
        else
            throw new InvalidActionException ();
        }
    return null;
}


/**
 * @param mimeTypeFilePath the file path to the Internet Mime Types file
 *                          which contains mappings from mime types to file extensions
 */
private static void loadMimeTypeFile (String mimeTypeFilePath) throws
    FileNotFoundException, IOException
{
    mimeTypeMap = new Properties ();

    File file = new File (mimeTypeFilePath);

    if (!file.exists())
        throw new FileNotFoundException();

    FileReader fis = new FileReader (file);

    BufferedReader reader = new BufferedReader (fis);

    String line = reader.readLine();
    String mimeType = null, extension = null;

    while (line != null)
    {
        line = line.trim();

        if (!line.startsWith("#") && line.length() > 0)
        {
            StringTokenizer st = new StringTokenizer (line);

            mimeType = st.nextToken();

            while (st.hasMoreTokens())
            {
                extension = st.nextToken();
                mimeTypeMap.put (extension, mimeType);
            }
        }

        line = reader.readLine();
    }

    fis.close();
}


}
```

```
package thinairapps.groupware.storeproviders.file;

import com.thinairapps.platform.provider.*;

import java.util.Properties;
import java.util.Enumeration;

/**
 * Defines a context object for the IMAP provider.
 *
 * @author meyerwil
 */
public final class FileStoreProviderContext extends StoreProviderContext
{
    // Data members
    protected static Properties  m_props          = null;

    private static Properties basePathStore;

    private static final String USER_PROP_KEY = "UserDirectory:";

    /**
     * Retrieves the object set supported by this provider -- the Groupware object set.
     *
     * @return A GroupwareObjectSet instance.
     *
     * @author meyerwil
     */
    public StoreProviderType getType ()
    {
        return new StoreProviderType ("thinairapps.groupware.api");
    }

    /**
     * Retireves a ContextProperties object containing user-editable required and optional
     * properties.
     *
     * @return ContextProperties object containing user-editable required and optional
       properties.
     *
     * @author meyerwil
     */
    public ContextProperties getProps ()
    {
        Properties optional;
        Properties required;

        optional = new Properties ();
        required = new Properties ();

        return new ContextProperties (optional, required);
    }

    /**
     * Tells the context to update its property set.  It will throw a
       SPInvalidContextPropsException
     * if it does not accept the properties.  This provider does require an SMTP server so it
       will
     * throw the exception if the user didn't specify one.
     *
     * @param props The new set of properties to commit.
     *
     * @author meyerwil
     */
    public void updateProps (Properties props)
    {
      m_props = props;

        Enumeration keys = m_props.keys();
        String user, dir;

        basePathStore = new Properties ();

        while (keys.hasMoreElements())
        {
            user = (String)keys.nextElement();

            if (user.startsWith(USER_PROP_KEY))
            {
                dir = m_props.getProperty (user);

                user = user.substring (USER_PROP_KEY.length());

                basePathStore.put (user, dir);
            }
        }
    }

    public static String getUserDirectory (String name)
    {
        String path = basePathStore.getProperty (name);

        if (path != null)
            return path;
        else
            return m_props.getProperty ("DefaultBasePath");
    }


    /**
     * Determines if this context has optional user-editable properties.  This context does
     * -- the LDAP server.
     *
     * @return Boolean true to indicate that this context does have optional properties.
     *
     * @author meyerwil
     */
    public boolean hasOptionalProps ()
    {
        return false;
    }

    /**
     * Determines if the context has required user-editable properties.  This context does
        -- the
     * SMTP server.
     *
     * @return Boolean true to indicate that this context does have required properties.
     *
     * @author meyerwil
     */
    public boolean hasRequiredProps ()
    {
        return true;
    }

    /**
     * This method indicates product information for the provider.
     *
     * @return A StoreProviderInfo object containing product information.
     *
     * @author meyerwil
     */
    public StoreProviderInfo getInfo ()
    {
        return new StoreProviderInfo (FileStoreProvider.MANUF_NAME,
                                      FileStoreProvider.MANUF_CONT,
                                        FileStoreProvider.APP_NAME,
                                        FileStoreProvider.VERSION,
                                        FileStoreProvider.BUILD,
                                        FileStoreProvider.APP_RELEASED);

    }
}
```

```
package thinairapps.groupware.storeproviders.file;

import java.util.StringTokenizer;

public class HTMLRipper
{
     private final static String SPECIAL_CHARS[] = { "&nbsp;",
                                                      "&amp;",
                                                      "&#174",
                                                      "&#38;",
                                                      "&#183", // no semicolon
                                                      "&#46;",
                                                      "&",
                                                      "&gt;",
                                                      "&lt;",
                                                      "$",
                                                      "\n",
                                                      "\r"};

    /**
     * remove anything that starts with a < and ends with a >
     * @param htmlCode source HTML
     */
    public static String removeTags (String htmlCode)
    {
        StringBuffer results = new StringBuffer();

        StringTokenizer st = new StringTokenizer(htmlCode,"<");

        String text = null;

        while(st.hasMoreTokens()) {

            text = st.nextToken();
            text = text.substring(text.indexOf(">")+1);

            if (text.length() > 0 && !text.equals("\n") && !text.equals(" "))
                results.append(text + " ");
        }

        return results.toString();
    }

    /**
     * remove anything that starts with a < and ends with a >
     * @param htmlCode source HTML
     */
    public static String removeTagsExcept (String htmlCode, String[] tags)
    {
        StringBuffer results = new StringBuffer();

        StringTokenizer st = new StringTokenizer(htmlCode,"<");

        String text = null;
        String tag = null;
        while(st.hasMoreTokens()) {

            text = st.nextToken();

            tag = text.substring(0,text.indexOf(">")).trim().toLowerCase();

            for (int i = 0; i < tags.length; i++)
            {
                if (tag.startsWith (tags[i]))
                    results.append ("<" + tag + ">");
            }

            text = text.substring(text.indexOf(">")+1);
            if (text.length() > 0 && !text.equals("\n") && !text.equals(" "))
                results.append(text);
        }

        return results.toString();
    }

    /**
     * remove the text specified in SPECIAL_CHARS
     */
    public static String removeSpecial(String htmlCode) {

        int numSpecial = SPECIAL_CHARS.length;
        int indexes[] = new int[numSpecial];

        int len = htmlCode.length();
        int newLen = len;

        char dummy = (char) 0;
        char buf[] = htmlCode.toCharArray();

        int i, j, k, index;

    outer:
        while (true) {
            k = 0; // count how many SPECIAL_CHARS have been found

        inner:
            for (i = numSpecial; --i >= 0; ) { // look through all the SPECIAL_CHARS

                if (indexes[i] == -1) { // no more SPECIAL to be found
                    if (++k == numSpecial) // have ALL SPECIAL been found?
                        break outer;
                    else // skip and keep LOOKING
                        continue inner;

                } else {
                    // look for more of SPECIAL
                    index = htmlCode.indexOf(SPECIAL_CHARS[i], indexes[i]);

                    if (index == -1) { // no more SPECIAL
                        indexes[i] = index; // mark as all done
                        continue inner; // continue to next
                    }
                }

                // replace all chars in SPECIAL with dummy char
                for (j = SPECIAL_CHARS[i].length(); --j >= 0; ) {
                    buf[ index + j ] = dummy;
                    --newLen;
                }

                indexes[i] = index + 1; // advance indexes[i] to avoid repeats
            }
        }


        // FIXME FIXME FIXME
        // use newLen and array-based implementation
        //
        StringBuffer sb = new StringBuffer(newLen);

        // copy all non-dummy chars into the return array
        for (i = 0; i < len; i++) {
            if (buf[i] == dummy)
                continue;
            else
                sb.append(buf[i]);
        }
        return sb.toString().trim();

    }


    /**
     * remove pairs of opening and closing tags
     */
    public static String removeTagPairContent (String htmlCode, String start, String end) {

        int i = 0;
        int endIdx = 0;

        while((i = htmlCode.indexOf(start,i)) > 0) {
            endIdx = htmlCode.indexOf(end,i);
            if (endIdx == -1)
                break;
            htmlCode = htmlCode.substring(0,i)+htmlCode.substring(endIdx+end.length(),      ↙
                htmlCode.length());
            i = htmlCode.indexOf(start,i);
        }

        return htmlCode;
    }


    /**
     * remove all script tags from HTML
     * @param htmlCode source HTML
     */
    public static String removeJS (String htmlCode) {

        htmlCode = removeTagPairContent(htmlCode, "<script", "</script>");
        htmlCode = removeTagPairContent(htmlCode, "<!-", "->");

        return htmlCode;

    }

        /**
     * remove all blank lines and new lines
     * @param text source String
     */
    public static String removeBlankLines(String text) {

        StringBuffer output = new StringBuffer ();

        StringTokenizer st = new StringTokenizer (text,"\n");

        String line;

        if (!st.hasMoreTokens()) {
            output.append(text);

        } else {

            while(st.hasMoreTokens()) {
                line = st.nextToken();
                line = line.trim();

                if (line.length() > 0)
                    output.append(line + "\n");
            }
        }

        return output.toString();
    }
    public static String processPage (String pageText)
    {

        pageText = removeJS(pageText);
        pageText = removeTags(pageText);
        pageText = removeBlankLines(pageText);
        pageText = removeSpecial(pageText);

        return pageText;
    }


}
```

==

TextFile Sample Groupware Provider

Wireless SDK for ThinAir Server

==

About This Sample

This sample Groupware Provider is compatible with any of the ThinAir Groupware Connectors included in the standard server installation. It demonstrates a simple Provider that reads data out of a text file data store, caches it, and services requests from the cache. The Provider currently ignores all of the request parameters and returns all messages in the cache with every request. It could easily be modified to pay attention to the 'max' member of the ItemRequest or the request bounds and filter the returned data in some way. Consult the ThinAir Platform API for information on the UserDataRequest / UserDataResponse protocol.

Requirements

This sample requires the following SDK JARs:

* platform.jar
* groupware.jar

This sample does not require any other external APIs.

Sample Files

This sample consists of the following file tree:

provider_integrated.ini and provider_standalone.ini - two possible config files, one of which should be copied into provider.ini

sourceFile.dat - a delimited text data file

sourceFileKey.txt - template for the data file format

TextFileProvider.jar - compiled Java code

\src - java source files

Building the Sample

Compile the sample code using the Java compiler of your choice. Make sure to append the required jar files above into your CLASSPATH.

Install the compiled sample code and provider.ini configuration file into a subdirectory of the ThinAir Server's \Providers subdirectory, given a name of your choice. If you are running TextFileProvider as an in-memory StoreProvider copy provider_integrated.ini into Providers/provider.ini. If you are running TextFileProvider as a standalone StoreProvider, copy provider_standalone into Providers\provider.ini before you begin. Within provider.ini under the [Provider Settings] heading, the SourceFile setting should point to the installation directory of the sourceFile.dat file on the machine hosting the Provider.

Start the ThinAir Server; it will load the sample code and begin executing it.

==

```
                    Last updated: 11.17.2000

              Copyright 1999, 2000 ThinAirApps Inc

=============================================================================
<username>|<password>
# Each message is encoded in a block like the one below
<sender's display name>|
<sender's email address>|
<sent date in  M-d-y h:mm a  format>|
<subject>|
<body>
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Thinairapps Imports
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.*;

import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.exception.*;

import com.thinair.utils.*;

//Standard Java Imports
import java.text.*;
import java.util.*;
import java.net.*;
import java.io.*;


/**
 * @(#)TextFileProvider.java
 *
 * Read a PIPE-delimited text file containing encoded messages
 * and deliver them to a Connector as Groupware objects
 */
public class TextFileProvider implements StoreProvider
{
    private static int counter = 0;

    private SupportedItems supportedItems;
    private String username, password;
    private Vector messages;
    private SimpleDateFormat dateFormatter;
    private int instanceNumber;

    //Error code from the ThinAir Groupware Access application.
    //Provider writers may define error codes for their own applications
    public static final int ERROR_AUTHENTICATION_FAILED = 4310;


    /**
     * build a new TextFileProvider
     * for the simple web provider there is nothing to do here
     */
    public TextFileProvider()
    {
        instanceNumber = ++counter;
        System.out.println("Starting TextFileProvider "+instanceNumber+"...");
        dateFormatter = new SimpleDateFormat("M-d-y h:mm a");

        // this cache holds all the messages in the source file
        messages = new Vector();
    }



    /**
     * Load the source file
     * Parse its contents into a username, password, and Message objects
     * authenticate the user's login based on username/password
     *
     * @return set of supported items
     */
    public SupportedItems connectUser(StoreProviderLogin login, StoreProviderContext     ↙
        context) throws ProviderException
    {
        try
        {
            // this will also obtain username and password info from the file
            loadSourceFile(((TextFileProviderContext) context).props);
        }
        catch (Exception e)
        {
            System.out.println("Caught an exception logging in: " + e.getMessage());

            //In a actual application, you would define your own error code
            throw new ProviderException(e.getMessage(), ProviderException.NO_ERROR_CODE);
        }

        // authenticate login
        if (! authenticateUser(login))
            //In an actual application, you would define your own error code
            throw new AuthenticationFailedException(ERROR_AUTHENTICATION_FAILED);

        // prepare the return object
        supportedItems = new SupportedItems();
        String name = TextFileProviderContext.APP_NAME;
        String location = null;

        try
        {
            location = InetAddress.getLocalHost().getHostAddress();
        }
        catch (Exception e)
        {
            location = "localhost";
        }

        // support no actions
        short actions[] = new short[0];
        SupportedItem item = new SupportedItem(ItemTypes.MESSAGE, name, location, actions);
        supportedItems.addItem(item);

        // return the completed SupportedItem set
        return supportedItems;
    }



    /**
     * disconnect a user - nothing to do for this simple Provider
     *
     */
    public void disconnectUser()
    {
        System.out.println("TextFileProvider "+instanceNumber+" shutting down.");
    }



    /**
     * not implemented for this simple Provider
     */
    public UserDataLocations getLocations(UserDataLocationRequest req)
    {
        UserDataLocations locs = new UserDataLocations(ItemTypes.MESSAGE, new String[0]);
        return locs;
    }



    /**
```

```
     * UserDataActions tell the provider to modify the backend data store in some way
     * This simple Provider does not support any actions
     *
     * @param action describes the requested action
     */
    public UserDataActionResponse doUserDataAction(UserDataAction action)
    {
        return null;
    }



    /**
     * The source file was loaded and cached in connectUser()
     * Service requests for messages here
     *
     * @param request represents the UserData request object
     */
    public UserData getUserData(UserDataRequest request)
    {
        ItemRequest itemReq = request.requests[0];

        // prepare the return object
        UserData ud = new UserData();
        ud.responses = new ItemRequestResponse[ 1 ];
        ud.responses[0] = new ItemRequestResponse();
        ud.responses[0].request = itemReq;

        short type = itemReq.itemType;

        // verify that the request is of the correct itemType
        if (type != ItemTypes.MESSAGE)
            throw new RuntimeException("Unknown item request type: "+type);


        // load the responses[0].items Vector with Message objects from the messages cache
        ud.responses[0].items = new StoreItems();

        // ignore bounds and max
        // return entire cache every time
        int numMessages = messages.size();
        for (int i = 0; i < numMessages; i++)
            ud.responses[0].items.addElement( messages.elementAt(i) );

        // N.B. if we had implemented bounded requests by filling in the responses[0].bounds ↙
            array
        // we could ask for a subset of the cached messages, and page back and forth within  ↙
            the
        // cache, retrieving the first 5, then the next 5, ect.

        // for now, make the Connector assume that the entire cache is returned each time
        ud.responses[0].items.setIsLastAvailable(true);

        // return completed response object
        return ud;
    }



    /*
     * Read and parse the source file
     * Determine username and password
     * Generate Message objects
     */
    private void loadSourceFile(Properties props) throws Exception
    {
        //Get the SourceFile value from provider.ini
        String sourceFile = props.getProperty("SourceFile");
        //Separate the sourcePath into filepath and fileName
        //look for the last instance of the \ String
        String lastDivider = "\\";

        //Get the divider's index in the String
        int lastIndex = sourceFile.lastIndexOf(lastDivider);

        //Get the sourcePath
        String sourcePath = sourceFile.substring(0,lastIndex+1);

        //Get the file name
        String fileName = sourceFile.substring(lastIndex+1,sourceFile.length());

        byte buf[] = null;

        //File file = new File(sourceFile);
        File file = new File(sourcePath, fileName);

        buf = new byte[ (int) file.length() ];

        try
        {
            FileInputStream fis = new FileInputStream( file );

            while (fis.read(buf) != -1) ;
            fis.close();
        } catch (Exception e) {
            throw new Exception("Cannot find / read source file: "+file.getAbsolutePath());
        }

        System.out.println("TextFileProvider "+instanceNumber+" loading data from "+file.
            getAbsolutePath());

        // buf now contains full file data
        String content = new String(buf);

        // use the Tokenizer to parse the data into Message objects
        parseSourceFile(content);
    }



    /*
     * Parse the source data
     * Extract username / password
     * Generate Message objects for the rest
     */
    private void parseSourceFile(String content) throws Exception
    {
        Tokenizer tok = new Tokenizer(content, '|');

        username = tok.nextToken();
        password = tok.nextToken();

        Message msg = null;
        GroupwareUserSMTPAddress address;
        while (tok.hasMoreTokens())
        {
            // displayName, email address
            msg = new Message();
            address = new GroupwareUserSMTPAddress(tok.nextToken(), tok.nextToken());

            msg.setFrom(address);
            msg.setReceivedDate( dateFormatter.parse( tok.nextToken() ) );
            msg.setSubject( tok.nextToken() );
            msg.setBody( tok.nextToken() );

            // add to global messages Vector
            messages.addElement(msg);
        }
        System.out.println(messages.size()+" Messages parsed by TextFileProvider "+
            instanceNumber);
    }



    /*
     * authenticate the login credentials passed to connectUser()
     * by checking against username/password stored in source file
     */
    private boolean authenticateUser(StoreProviderLogin login)
    {
        if (! login.name.equals( username )) return false;
        if (! login.password.equals( password )) return false;

        System.out.println("TextFileProvider "+instanceNumber+" authenticates "+username);
        return true;
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Thinairapps Imports
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.*;

import com.thinairapps.platform.provider.*;
import com.thinairapps.platform.exception.*;

import com.thinair.utils.*;

//Standard Java Imports
import java.util.*;



/**
 * @(#)TextFileProvderContext
 *
 * provides static information for and about the TextFileProvider
 */
public class TextFileProviderContext extends StoreProviderContext
{

    // Version info
    protected static final String VERSION       = "1.2";
    protected static String APP_NAME            = "TextFileProvider";
    protected static final String MANUF_NAME    = "ThinAirApps";
    protected static final String MANUF_CONT    = "www.ThinAirApps.com";
    protected static final String   BUILD        = "1";
    protected static final Date   APP_RELEASED = new Date ();

    protected Properties props;



    /**
     * Determines if the context has optional user-editable properties. Implementors should
     * return true if they can offer optional Properties to the user, but do not require
       these
     * properties to be set in order to correctly serve user connections.  StoreProviders may
       have
     * both property types.
     *
     * @return A boolean indicating whether or not the context has optional properties.
     */
    public boolean hasOptionalProps() { return false; }



    /**
     * @return ProviderObjectSet indicating the friendly and class names of StoreItem
       subclasses
     *         understood by this StoreProvider.
     */
    public StoreProviderType getType() { return new StoreProviderType("thinairapps.groupware.
        api"); }



    /**
     * Called by a client to ask for product information on the provider.
     * @return StoreProviderInfo containing information on this provider.
     */
    public StoreProviderInfo getInfo ()
    {
        return new StoreProviderInfo ( MANUF_NAME,
                                       MANUF_CONT,
                                       APP_NAME,
                                       VERSION,
                                       BUILD,
                                       APP_RELEASED );
    }



    /**
     * Tells the context to update its property set.  It will throw a                ↙
        SPInvalidContextPropsException
     * if it does not accept the properties.
     * @param props The new set of properties to commit.
     */
    public void updateProps(Properties p)
    {
        if (p.getProperty("SourceFile") == null)
            throw new RuntimeException("Cannot find property SourceFile in provider.ini");

        props = p;
    }



    /**
     * @return A boolean indicating whether or not the context can offer required properties.
     */
    public boolean hasRequiredProps() { return true; }



    /**
     * Retireves a ContextProperties object containing user-editable required and optional
     * properties
     * @return ContextProperties object containing user-editable required and optional     ↙
        properties.
     */
    public ContextProperties getProps()
    {
        Properties required = (props == null) ? new Properties() : (Properties) props.clone   ↙
            ();
        return new ContextProperties( new Properties(), required);
    }
}
```

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Standard Java Imports
import java.util.*;



/**
 * @(#)Tokenizer.java
 *
 * Simple tokenizer that, unlike StringTokenizer, will return an empty
 * String when two tokens appear right next to each other
 */
public class Tokenizer
{

    private static final String BLANK = "";
    private char DELIM;
    private int currentPosition;
    private int maxPosition;
    private String str;



    /**
     * buld a new Tokenizer for the given String with the given delimiter
     * @param s source String
     * @param d delimiter character
     */
    public Tokenizer(String s, char d)
    {
      str = s;
      DELIM = d;
      currentPosition = 0;
      maxPosition = str.length();
    }



    /**
     * Returns the next token from this string tokenizer.
     *
     * @return      the next token from this string tokenizer - may be "" if two
     *              DELIMs next to each other appear in the String
     * @exception  NoSuchElementException  if there are no more tokens in this
     *              tokenizer's string.
     */
    public String nextToken()
    {
        int start = currentPosition;
        if (++currentPosition > maxPosition) throw new NoSuchElementException();

        if (currentPosition == maxPosition || (str.charAt(currentPosition) == DELIM))
            return BLANK;

        // else....
        if (str.charAt(start) == DELIM) ++start;
        while ((currentPosition < maxPosition) && (str.charAt(currentPosition) != DELIM))
            currentPosition++;

        return str.substring(start, currentPosition).trim();
    }
    /**
     * Returns the same value as the <code>hasMoreTokens</code>
     * method. It exists so that this class can implement the
     * <code>Enumeration</code> interface.
     *
     * @return  <code>true</code> if there are more tokens;
     *          <code>false</code> otherwise.
     * @see     java.util.Enumeration
     * @see     java.util.StringTokenizer#hasMoreTokens()
     */
    public boolean hasMoreTokens()
    {
        return (currentPosition < maxPosition);
    }
}
```

```
===============================================================================

                    Send Email Sample Groupware Connector

                      Wireless SDK for ThinAir Server

===============================================================================


-----------------
About This Sample
-----------------

This sample Groupware Connnector is compatible with any of the ThinAir
Groupware Providers included in the standard server installation.  It
demonstrates how to write a simple Connector using the ThinAir Groupware API
that allows a user to create and send an email message.

The Send Email Connector first displays a UI prompting the user for the
from, subject, and body elements of the email message.  It then reads the
user's login credentials from the connector.ini file.  The Connector could
easily be modified to prompt for this information, as the GetItems Groupware
Connector does.  It then connects to the message store, creates a Groupware
Object representing the email message, and sends it.  It then logs the user
off.  This example could easily be modified to create items of other types,
e.g. Event or Task.

N.B. This sample Connector is written for WML devices ONLY.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * groupware.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

     connector.ini - connector configuration file

     SendItemsConnector.class - compiled Java code

     /src - java source files


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  The included
MAKE script will compile the sample using the JDK, if available.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the Send Email Groupware Connector
has been loaded and initialized.  From a WML device, enter the IP address
```

listed as the value for ApplicationPath in connector.ini (your ThinAirServer IP address), followed by /samples/sendemail. For a machine with IP address 111.222.12.34 this would be:

    http://111.222.12.34/samples/sendemail

Follow the on-screen instructions.

```
===============================================================================
                          Last updated: 11.13.2000
                    Copyright 1999, 2000 ThinAirApps Inc.
===============================================================================
/**
 * @(#)SendEmailConnector.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

//core ThinAir Server API functionality

import com.thinairapps.platform.device.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.provider.*;

//rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

// the groupware packages
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.* ;

import java.util.*;
import java.io.*;


/**
 * This is a simple sample whose purpose is to illustrate the use of the ThinAir Groupware ↙
   Library for creating groupware items.
 * This sample renders WML. It prompts the user to enter to address, a subject and a body, ↙
   and then procedes
 * to log them onto their message store, create (send) the item and then log them off again. ↙
   This sample could easily be
 * modified to create items of other types, eg. Event or Task
 *
 * For a comprehensive reference of the Groupware Library see the ThinAir Groupware javadocs.
 */
public class SendEmailConnector implements Connector
{

    //The friendly name of this sample app
    protected String          appName;

    //Our access point to the services of ThinAir Server
    protected ConnectorAccess connectorAccess;

    protected String provider;
    protected String userName;
    protected String password;
    protected String host;
    protected String fromDisplay;
    protected String fromEmail;

    private final static String SUPPORTED_ITEMS_CACHE_KEY = "supports";



    /**
     * init() is called by the ThinAirServer when the Connector is loaded.  It provides the ↙
       connector with
     * resources it needs to interact with the ThinAirServer.
     * For more information about the Connector interface, see the javadocs for the ThinAir ↙
       Server API
     *
     * @param applicationName is a String derived from connector.ini.
     * @param applicationPath is a String dervid from connector.ini.  We don't need this for ↙
       this sample.
     * @param connectorProps is a Properties list containing developer assigned ↙
       connector-specific properties.
     * @param connectorAccess is our access point to the services provided by ThinAir Server.
     * @param ApplicationLog is used for logging
     */
    public void init(String applicationName, String applicationPath,
                     Properties connectorProps, ConnectorAccess connectorAccess, com. ↙
                        thinairapps.platform.connector.ApplicationLog al) throws ↙
                        ConnectorInitException {
        this.appName = applicationName;
        this.connectorAccess = connectorAccess;

        // get the all required variables from the properties list (connector.ini). make sure↙
            you set this up before
        //running this example
        provider = connectorProps.getProperty("Provider");

        // your user name
        userName = connectorProps.getProperty("UserName");

        // your password
        password = connectorProps.getProperty("Password");

        // the ip of your message host, note: if using IMAP or POP
        // the ip of the smtp server you will be using is in the providers provider.ini file
        host = connectorProps.getProperty("Host");

        // your display name
        fromDisplay = connectorProps.getProperty("FromDisplay");

        // your email address
        fromEmail = connectorProps.getProperty("FromEmail");

        if (provider.length() == 0 || userName.length() == 0 || password.length() == 0 ||
            host.length() == 0 ||fromDisplay.length() == 0 || fromEmail.length() == 0)
            throw new ConnectorInitException("connector.ini must have entries for Provider, ↙
                UserName, Password, Host, FromDisplay, and FromEmail");
    }


    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing ↙
       the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used ↙
       to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
       to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample ↙
       connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this ↙
       Connector supports.
     */
    public String[] getDevices()
    {
        String deviceType = "TA_WAP";
        String[] deviceTypes = {deviceType};
        return deviceTypes;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming    ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,    ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters   ↙
        from the device.
     * @param device a Device object created in the image of the actual device making this   ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream result) throws          ↙
        IOException {

        String resultString = null;

        //get the 'action' parameter from the request. This is an HTTP param we define to    ↙
            determine what action
            //to take when we get a request.
        String action = props.getProperty("action");

        String sessionId = null;

        if (action == null)
        {
            // if this is the first hit (or any request for the main deck)
            // build a deck that lets the user enter information.
            resultString = renderCreateMesage();

        } else if (action.equals("createMsg"))
        {
            //  they have already entered the information, get the messages
            String to;
            String sub;
            String body;

            // get the required parameters

            // the mail host we will be accessing fom the HTTP params
            // the recipient
            to = props.getProperty("toInput");

            // the message subject
            sub = props.getProperty("subInput");

            // the message body
            body = props.getProperty("bodyInput");

            try
            {
                // we have all the information we need to logon and send the message
                // log you in to teh message store
                sessionId = loginUser(provider, host, userName, password);
                // get the default location from the supported Items returned from Login that↙
                    we stored in the session cache
                String defaultLocation = getDefaultLocation(ItemTypes.MESSAGE, sessionId);

                // send the message
                sendMessage (defaultLocation, sessionId, fromDisplay, fromEmail, to, sub,    ↙
                    body);

                // log them off
                connectorAccess.getStoreProvider(sessionId).disconnectUser();

                //  then delete the session
                connectorAccess.deleteSession(sessionId);

                // now render a screen indication success
                resultString = renderSuccess("Message Successfully Created");

            } catch (Exception e)
            {
                // need to do error checking here, we simply display the error message and   ↙
                    provide a link
                // back to the welcome screen, a larger app would handle each error          ↙
                    seperatley and navigate the
                // user accordingly
                resultString = renderException(e.getMessage ());
            }

        }

        byte[] resultBytes = resultString.getBytes();
        result.write(resultBytes);
    }



    /**
     * This method logs the user in to their message store
     *
     * @param providerName the name of the provider being used to access the message store.
     * @param host the ip or server name of the message store.
     * @param userName the user name of the account being logged onto.
     * @param password the password for this user.
     * @return a providerSessionID if success, otherwise an error will be thrown.
     */
    protected String loginUser (String providerName, String host, String userName,
                                String password) throws Exception {

        String SID = null;
        try {
            // Create a new Session with the specified provider and returns a unique Session ↙
                ID.
            SID = connectorAccess.createProviderSession (providerName);

            // Get the StoreProviderProxy associated with the session we just created,
            // this is what is used to interact with the Provider
            StoreProviderProxy spProxy =  connectorAccess.getStoreProvider (SID);

            // Create a StoreProviderLogin object, this defines the action the Provider will ↙
                execute
            StoreProviderLogin login = new StoreProviderLogin (userName, password, host);

            // use the StoreProviderProxy to login. The provider returns the items it        ↙
                supports and the actions on them
            SupportedItems supports = spProxy.connectUser (login);

            // we cache the supported Items because we know we will need them later
            connectorAccess.getSessionCache(SID).put(SUPPORTED_ITEMS_CACHE_KEY,supports);
        } catch (NoSuchProviderException e) {
            throw new Exception("No Provider named "+providerName+" was loaded by the ThinAir↙
                Server");

        }

        return SID;
    }



    /**
     * This method sends a message. It assumes you are in the logged in state
     *
     * @param location the location the message is to be created in, this does not matter    ↙
        when creating a message, however if creqating
     *                  a post or other item we could specify the location the item be placed↙
        in.
     * @param SID a valid session Id, the user must already be authenticated.
     * @param fromDisplay the items storeIndex to start at, this may be null if starting at  ↙
        the beginning
     * @param fromEmail the maximum number of messages to retrieve.
     * @param to the recipient of the message you are sending. This can be their email       ↙
        address, or you can have the server try and identify them either throught LDAP if    ↙
        supported and activated or through the message stores internal name resolution system
     * @param sub the subject of the message you are sending.
     * @param body the body of the message you are sending.
     */
    protected void sendMessage (String location, String SID, String fromDisplay, String       ↙
        fromEmail,
                                  String to, String sub, String body) throws Exception
    {

        Message msg = new Message();

        GroupwareUserSMTPAddress fromAddr = new GroupwareUserSMTPAddress (fromDisplay,        ↙
            fromEmail);
        msg.setFrom (fromAddr);

        // the recipients are always an array of GroupwareUserSMTPAddresss' we are assuming  ↙
            for this example that there will only be one
        // feel free to tokenize a list of comma or semi-colon delimited list of recipients  ↙
            here
        GroupwareUserSMTPAddress[] toAddrs = new GroupwareUserSMTPAddress[1];
        if (to.indexOf("@") > 0)
            // it is a valid email address
            toAddrs[0] = new GroupwareUserSMTPAddress(null, to);
        else
            // not a valid address, leave it up to the Provider or mesasge store to validate
            toAddrs[0] = new GroupwareUserSMTPAddress(to, null);

        msg.setRecipients(Message.RecipientType.TO, toAddrs);

        // if we suppported cc
        /*
        GroupwareUserSMTPAddress[] ccAddrs = new GroupwareUserSMTPAddress[1];
        if (to.indexOf("@") > 0)
            ccAddrs[0] = new GroupwareUserSMTPAddress(null, cc);
        else
            ccAddrs[0] = new GroupwareUserSMTPAddress(cc, null);

        msg.setRecipients(Message.RecipientType.CC, ccAddrs);
        */

        msg.setSubject(sub);
        msg.setBody (body);

        // the location the message is to be created in, this does not matter when creating a↙
            message, however if creating a
        // post or other item we could specify the location the item be placed in.
        msg.setLocationInStore (location);

        StoreProviderProxy spProxy = connectorAccess.getStoreProvider (SID);

        AddNewGroupwareItem addAction = new AddNewGroupwareItem (msg);
        spProxy.doUserDataAction (addAction);


        return;
    }



    /**
     * A utility method that simply looks in the session cache for the SupportedItems,   ↙
        extracts the
     * default location for the specified item type and returns it
     *
     * @param SID a valid session Id.
     * @param itemtype the type of item.
     * @return the default location.
     */
    private String getDefaultLocation (int itemType, String SID) throws                  ↙
        NoSuchSessionException
    {
        // look for SupportedItems in the cache
        SupportedItems sis = (SupportedItems)connectorAccess.getSessionCache (SID).get   ↙
            (SUPPORTED_ITEMS_CACHE_KEY);
        Enumeration elem = sis.getItems();
        SupportedItem si = null;

        // cycle through them until we find the type we need and return it
        while (elem.hasMoreElements())
        {
            si = (SupportedItem)elem.nextElement();
            if (itemType == si.getType())
            {
                return si.getDefaultLocation();
            }
        }

        //itemType not found, this should never happen
        return null;
    }


    /**
     * This method renders a deck with several cards including a welcome card and card for   ↙
        entering information...
     *
     * @return the rendered deck.
     */
    private String renderCreateMesage()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create the first card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");
        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_WRAP);


        p.addChild(new Text("Send Email"));
        p.addChild(new Break());
        p.addChild(new Text("Sample Connector"));
        //add the Paragraph to the card
        card1.addParagraph(p);
        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_WRAP);

        //a link to the second card
        String href = "?#compose";

        //make a Go task with the href
        Go go = new Go(href,true,Go.METHOD_GET);

        //create the Anchor with the Go task
        Anchor anchor = new Anchor(go,new Text("Login"));

        //add the anchor to the Paragraph
        p.addChild(anchor);

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the first card to the deck
        deck.addCard(card1);

        LabeledInput toInput = new LabeledInput("to", "TO:");
        LabeledInput subInput = new LabeledInput("sub", "SUBJECT:");
        LabeledInput bodyInput = new LabeledInput("body" ,"BODY:");

        LabeledInput[] inputs = {toInput, subInput, bodyInput};

        //set the URL params to the values in the WML variables &amp;, the escape sequence
            for ampersand, delimits name-
        //value pairs.  $ is used to dereference a WML variable. A random number is used so
            the next time you get to this page you will actually
        // be hitting the server rather that retrieving from the phone's cache
        href = "?action=createMsg&amp;toInput=$(to)&amp;subInput=$(sub)&amp;bodyInput
            =$(body)&amp;rnd="+Math.random();

        MultipleInputCard mic = new MultipleInputCard("compose");


        mic.buildCard(href,"Send",inputs,Go.METHOD_GET);

        deck.addCard(mic);

        String resultString = deck.render();

        return resultString;

    }


    /**
     * This is a simple exception rendering method.
     *
     * @param message the message to be presented to the user
     * @return the rendered WML deck
     */
    private String renderException (String message)
    {
        WMLTagDocument deck = new WMLTagDocument();
        DisplayCard card = new DisplayCard();
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());

        String href = "?rnd="+Math.random();
        Go go = new Go(href,true,Go.METHOD_GET);
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        p.addChild(anchor);
        card.addParagraph(p);
        deck.addCard(card);

        String resultString = deck.render();

        return resultString;
    }



    /**
     * This is a simple rendering method indicating success.
     *
     * @param message the message to be presented to the user
     * @return the rendered WML deck
     */
    private String renderSuccess (String message)
    {
        WMLTagDocument deck = new WMLTagDocument();
        DisplayCard card = new DisplayCard();
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());

        String href = "?rnd="+Math.random();
        Go go = new Go(href,true,Go.METHOD_GET);
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        p.addChild(anchor);

        card.addParagraph(p);
        deck.addCard(card);

        String resultString = deck.render();

        return resultString;
    }
}
```

```
==============================================================================

                  GetItems Sample Groupware Connector

                     Wireless SDK for ThinAir Server

==============================================================================


-----------------
About this Sample
-----------------

This sample Groupware Connnector is compatible with any of the ThinAir
Groupware Providers included in the standard server installation.  It
demonstrates how to write a simple Connector using the ThinAir Groupware API
that allows a user to log into their email store and retrieve messages.

The GetItems Connector prompts the user for their email host, username, and
password.  If the login is accepted by the Groupware Provider, the Connector
retrieves the first 5 messages in the user's inbox.  This example could easily
be modified to retrieve items of other types, e.g. Events or Tasks.

N.B. This sample Connector is written for WML devices ONLY.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * groupware.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    GetItemsConnector.class - compiled Java code

    /src - java source files


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  The included
MAKE script will compile the sample using the JDK, if available.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the Get Items Connector has been
loaded and initialized.  From a WML device, enter the IP address listed as the
value for ApplicationPath in connector.ini (your ThinAirServer IP address),
followed by /samples/getitems.  For a machine with IP address 111.222.12.34
this would be:

     http://111.222.12.34/samples/getitems
```

Follow the on-screen instructions.

```
==============================================================================
                          Last updated: 11.13.2000
                   Copyright 1999, 2000 ThinAirApps Inc.
==============================================================================
/**
 * @(#)GetItemsConnector.java
 *
 * Copyright(c)2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */

//core ThinAir Server API functionality
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.provider.*;

//rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

// the groupware packages
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.* ;

import java.util.*;
import java.io.*;

/**This is a simple sample whose purpose is to illustrate the use of the ThinAir Groupware ↙
   Library forn retrieving groupware Items.
 * This sample renders WML. It prompts the user to enter a mail host, a userName and a ↙
   password and then procedes
 * to log them onto their message store, retrieve the first 5 messages from the default ↙
   location and render the
 * headers. This sample could easily be modified to retrieve items of other types, eg. Event ↙
   or Task, and render
 * them.
 * For a comprehensive reference of the Groupware Library see the ThinAir Groupware javadocs.
 *
 */
public class GetItemsConnector implements Connector
{

    //The friendly name of this sample app
    protected String          appName;
    //Our access point to the services of ThinAir Server
    protected ConnectorAccess connectorAccess;

    //The provider
    protected String provider;

    private final static String SUPPORTED_ITEMS_CACHE_KEY = "supports";
    // Max number of characters to be displayed in the subject portion of the header
    private final static int HEADER_SUBJECT_LENGTH = 10;
    // Max number of characters to be displayed in the from portion of the header
    private final static int HEADER_FROM_LENGTH = 11;


    /**init() is called by the ThinAirServer when the Connector is loaded.  It provides the ↙
       connector with
     * resources it needs to interact with the ThinAirServer.<br><br>
     * For more information about the Connector interface, see the javadocs for the ThinAir ↙
       Server API
     *
     * @param applicationName is a String derived from connector.ini.
     * @param applicationPath is a String dervid from connector.ini.  We don't need this for ↙
        this sample.
     * @param connectorProps is a Properties list containing developer assigned             ↙
        connector-specific properties.
     * @param connectorAccess is our access point to the services provided by ThinAir Server.
     * @param applicationLog is used for Logging
     */
    public void init(String applicationName, String applicationPath, Properties             ↙
        connectorProps,
                     ConnectorAccess connectorAccess, com.thinairapps.platform.connector.   ↙
                        ApplicationLog al) throws ConnectorInitException
    {

        this.appName = applicationName;
        this.connectorAccess = connectorAccess;

        // get the provider name from the properties list (connector.ini)
        // You can modify this in connector.ini to access a different provider, to retrieve
        // items from a different mesage Store type, eg. IMAP or POP3
        provider = connectorProps.getProperty("Provider");

        if (provider.length() == 0) throw new ConnectorInitException("No Provider entry in   ↙
            connector.ini");
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a        ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing  ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used  ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this       ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String deviceType = "TA_WAP";
        String[] deviceTypes = {deviceType};
        return deviceTypes;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming     ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,    ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters    ↙
        from the device.
     * @param device a Device object created in the image of the actual device making this    ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream result) throws          ↙
        IOException
    {

        String resultString = null;

        //get the 'action' parameter from the request. This is an HTTP param we define to
        //determine what action to take when we get a request.
        String action = props.getProperty("action");

        String sessionId = null;

        if (action == null)
        {
            // if this is the first hit (or any request for the main deck)
            // build a deck that lets the user enter information.
            resultString = renderLoginPrompt();

        }
        else if (action.equals("msgs"))
        {
            //  they have already entered the information, get the messages
            String host;
            String userName;
            String password;

            // the mail host we will be accessing fom the HTTP params
            host = props.getProperty("host");

            // your user name
            userName = props.getProperty("usr");

            // and password
            password = props.getProperty("pwd");

            // make sure they have entered all information
            if (host==null || host.length ()==0 || userName==null || userName.length ()==0    ↙
                || password==null || password.length ()==0)
            {
                // if they haven't, tell them what they did wrong.
                resultString = renderException("You must enter all information");

            }
            else
            {
                // we have all the information we need to logon

                try
                {
                    sessionId = loginUser(provider, host, userName, password);

                    // get the default location from the supported Items returned from Login ↙
                        that we stored in the session cache
                    String defaultLocation;
                    defaultLocation = getDefaultLocation(ItemTypes.MESSAGE, sessionId);
                    /*
                        In this simple example we only retrieve items from the default        ↙
                            location, however ThinAir Groupware Providers provide
                        a means to query a store for the available locations, then use that   ↙
                            location to retrieve items from. This method can
                        be used to access Exchange public and private folders, as well as     ↙
                            IMAP.
                    // Get the locations
                    UserDataLocationRequest udLocReq = new UserDataLocationRequest ();
                    //udLocReq.itemType = itemTypeParamToCode (itemType);
                    udLocReq.itemType = ItemTypes.MESSAGE;
                    udLocReq.maxDepth = 0;  // how many folders deep do we want to go?, a↙
                        depth of more than 0 will include sublocation marked by the      ↙
                        '/'charactger
                    udLocReq.maxLocations = -1;
                    udLocReq.rootLocation = rootLocation; // null if starting from the   ↙
                        root.

                    UserDataLocations udLocations = connectorAccess.getStoreProvider     ↙
                        (sessionId).getLocations (udLocReq);

                    String locations[] = udLocations.getNames();

                    // now we have an array of other locations, you may use one of these ↙
                        to retrieve items from.

                */

                // get the messages
                StoreItems messages = getStoreItems(defaultLocation, sessionId, null, 5, ↙
                    ItemTypes.MESSAGE);

                // log them off
                connectorAccess.getStoreProvider(sessionId).disconnectUser();

                //  then delete the session
                connectorAccess.deleteSession(sessionId);

                // now render them
                resultString = renderMessageHeaders(messages);

                // in this example we logged on, got the messages then logged off again. ↙
                    A more complete app would
                // hold the session open between requests, and cache the retrieved      ↙
                    StoreItems in the session cache, using this to
                // feed item bodies out to the client without going to the provider each ↙
                    time

            }
            catch (Exception e)
            {
                // need to do error checking here, we simply display the error message   ↙
                    and provide a link
                // back to the welcome screen, a larger app would handle each error      ↙
                    separately and navigate the
                // user accordingly
                resultString = renderException(e.getMessage ());
            }
        }
    }

    byte[] resultBytes = resultString.getBytes();
    result.write(resultBytes);
}


/**
 * This method renders a deck with several cards including a welcome card and a card for ↙
    entering information...
 *
 * @param providerName the name of the provider being used to access the message store.
 * @param host the ip or server name of the message store.
 * @param userName the user name of the ccount being logged onto.
 * @param password the password for this user.
 * @return a providerSessionId if success, otherwise an error will be thrown.
```

```
     */
    protected String loginUser (String providerName, String host, String userName,
                                String password) throws Exception {

        String SID = null;

        try
        {
            // Create a new Session with the specified Provider and returns a unique Session ↙
                ID.
            SID = connectorAccess.createProviderSession (providerName);

            // Get the StoreProviderProxy associated with the session we just created,
            // this is what is used to interact with the Provider
            StoreProviderProxy spLite =  connectorAccess.getStoreProvider (SID);

            // Create a StoreProviderLogin object, this defines the action the provider will ↙
                execute
            StoreProviderLogin login = new StoreProviderLogin (userName, password, host);

            // use the providerProxy to login. The provider returns the itens it supports
            SupportedItems supports = spLite.connectUser (login);

            // we cache the supported Items because we know we will need them later
            connectorAccess.getSessionCache(SID).put(SUPPORTED_ITEMS_CACHE_KEY,supports);

        }
        catch (NoSuchProviderException e)
        {
            throw new Exception("No Provider named "+providerName+" was loaded by the ThinAir↙
                Server");
        }

        return SID;
    }


    /**
     * This method retrieves storeItems from the Provider. There is no cache involved here.
     * This same method can be used to retrieve any item type, starting at a particular item ↙
        id,
     * up to a max number.
     *
     * @param location the location or folder items are to be retrieved from.
     * @param SID a valid session Id, the user must already be authenticated.
     * @param startIdx the items storeIndex to start at, this may be null if starting at the ↙
        beginning
     * @param max the maximum number of messages to retrieve.
     * @param itemtype the type of item to retrieve.
     * @return the storeItems, this will be of length 0 if no items are retrieved, never null↙
        except if exception thrown
     */
    protected StoreItems getStoreItems (String location, String SID, String startIdx, int max↙
        , short itemtype) throws Exception
    {
        ItemRequest iReq = new ItemRequest ();
        iReq.itemType = itemtype;
        iReq.itemLocation = location;
        iReq.max = max;
        iReq.startID = startIdx;
        iReq.bounds = null;

        UserDataRequest udReq = new UserDataRequest ();
        udReq.requests = new ItemRequest[1];
        udReq.requests[0] = iReq;

        // getting items of another type we shoud check in the supportedItems to verify they ↙
            are supported by the current Provider
        StoreItems storeItems = (StoreItems)connectorAccess.getStoreProvider (SID).          ↙
```

```
            getUserData (udReq).responses[0].items;

        return storeItems;
    }



    /**
     * A utility method that simply looks in the session cache for the SupportedItems,
        extracts the
     * default location for the specified item type and returns it
     *
     * @param SID a valid session Id.
     * @param itemtype the type of item.
     * @return the default location.
     */
    private String getDefaultLocation (int itemType, String SID) throws
        NoSuchSessionException
    {
        // look for SupportedItems in the cache
        SupportedItems sis = (SupportedItems)connectorAccess.getSessionCache (SID).get
            (SUPPORTED_ITEMS_CACHE_KEY);
        Enumeration elem = sis.getItems();
        SupportedItem si = null;

        // cycle through them until we find the type we need and return it
        while (elem.hasMoreElements())
        {
            si = (SupportedItem)elem.nextElement();
            if (itemType == si.getType())
            {
                return si.getDefaultLocation();
            }
        }

        //itemType not found, this should never happen
        return null;
    }



    /**
     * This method renders a deck with several cards including a welcome card and card for
        entering login information...
     *
     * @return the rendered deck.
     */
    private String renderLoginPrompt()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create the first card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_WRAP);

        p.addChild(new Text("Get Items"));
        p.addChild(new Break());
        p.addChild(new Text("Sample Connector"));

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_WRAP);

        //a link to the second card
        String href = "?#c2";
        //make a Go task with the href
        Go go = new Go(href,true,Go.METHOD_GET);

        //create the Anchor with the Go task
        Anchor anchor = new Anchor(go,new Text("Login"));

        //add the anchor to the Paragraph
        p.addChild(anchor);

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the first card to the deck
        deck.addCard(card1);

        // create a new MultipleInputCard and give it the ID 'c2' This allows the user to ↙
            type in information
        MultipleInputCard card2 = new MultipleInputCard("c2");

        // host name input
        LabeledInput host = new LabeledInput("host","Host:");

        // set the input text to lowecase by deafult
        host.setFormat("*m");

        // username input
        LabeledInput userName = new LabeledInput("usr","Username:");
        userName.setFormat("*m");

        // pasword input
        LabeledInput password = new LabeledInput("pwd","Password:");
        password.setFormat("*m");
        //display input characters as stars...
        password.setType(Input.TYPE_PASSWORD);


        LabeledInput[] inputs = {host, userName, password};

        //set the URL params to the values in the WML variables &amp;, the escape sequence ↙
            for ampersand, delimits name-
        //value pairs.  $ is used to dereference a WML variable.
        href = "?action=msgs&amp;color=$(color)&amp;host=$(host)&amp;pwd=$(pwd)&amp;usr ↙
            =$(usr)&amp;rnd="+Math.random();


        //build the card with the href, "Submit" as the button label, the array of Inputs, ↙
            and the method specified.
        card2.buildCard(href,"Submit",inputs, Go.METHOD_GET);

        deck.addCard(card2);

        String resultString = deck.render();

        return resultString;
    }



    /**
     * This method renders a deck with one card containing the message headers if any, else ↙
        an error message explaining no messages available.
     *
     * @param msgs the messages to render.
     * @return the rendered deck.
     */
    public String renderMessageHeaders (StoreItems msgs)
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        if (msgs.size() == 0)
        {
            // no messages available for display
            DisplayCard card = new DisplayCard ("err1","ERROR");
            card.buildCard ("There are no messages in this folder",Paragraph.ALIGN_CENTER);
            deck.addChild (card);
        }
        else
        {
            // there are messages, go ahead and render them..
            String url = null;

            int msgIdx;
            String titleText = ((Message)msgs.elementAt (0)).getLocationInStore();
            if (titleText == null)
                titleText = "INBOX";

            //create the first card in the deck and give it the ID 'c1'
            DisplayCard card = new DisplayCard("c1");

            Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

            p.addChild(new Text (titleText));

            p.addChild(new Break());
            card.addParagraph (p);

            Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

            // add the message items
            Message msg;
            for (int i = 0; i < msgs.size (); i++)
            {
                msg = (Message)msgs.elementAt (i);

                // display an image if the phone supports it
                p2.addChild(new Image(new Icon(Icon.ENVELOPE1),Image.ALIGN_MIDDLE,null));

                //get the subject text, shorten if necessary
                String subText = "no subject";
                if (msg.getSubject() != null && msg.getSubject().length() > 0)
                {
                    subText = msg.getSubject();
                    if (subText.length() > HEADER_SUBJECT_LENGTH)
                        subText = subText.substring(0,HEADER_SUBJECT_LENGTH-3) + "...";

                    subText = ReservedCharacter.reformat (subText);
                }
                p2.addChild(new Text(subText));


                //get the from text, shorten if necessary
                String fromText = "no sender";
                if (msg.getFrom() != null)
                {
                    fromText = msg.getFrom().getDisplayName();
                    if (fromText == null)
                    {
                        fromText = msg.getFrom().getAddress();
                        if (fromText != null)
                        {
                            if (fromText.length() > HEADER_FROM_LENGTH)
                                fromText = fromText.substring (0, HEADER_FROM_LENGTH);
                        }
                        else
                            fromText = "no sender";
                    }
                    else if (fromText.length() > HEADER_FROM_LENGTH)
                        fromText = fromText.substring (0, HEADER_FROM_LENGTH);

                }
                p2.addChild(new Text(fromText));

                p2.addChild(new Break());
            }

            p2.addChild(new Break());


            // link home.
            String href = "?rnd="+Math.random();
            Go go = new Go(href,true,Go.METHOD_GET);
            Anchor anchor = new Anchor(go,new Text("Start again..."));

            p2.addChild(anchor);

            card.addParagraph(p2);
            deck.addCard (card);

        }

        return deck.render();
    }



    /**
     * This is a simple exception rendering method.
     *
     * @param message the message to be presented to the user
     * @return the rendered WML deck
     */
    private String renderException (String message)
    {
        WMLTagDocument deck = new WMLTagDocument();
        DisplayCard card = new DisplayCard();
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());


        String href = "?rnd="+Math.random();
        Go go = new Go(href,true,Go.METHOD_GET);
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        p.addChild(anchor);

        card.addParagraph(p);
        deck.addCard(card);

        String resultString = deck.render();

        return resultString;
    }

}
```

```
==============================================================================

                    CustomItem Sample Groupware Connector

                      Wireless SDK for ThinAir Server

==============================================================================


-----------------
About this Sample
-----------------

This sample Groupware Connnector demonstrates how to write a simple Connector
using the ThinAir Groupware API that allows a user to log in to their
groupware store, and retrieve an item with custom fields, or add a new item
with custom fields. Of the Providers shipped with the ThinAir Server, the
Domino and Exchange Providers both support custom item handling.

The CustomItem Connector first has the user log into the groupware store,
using login data entered in the connector.ini file.  The Connector can easily
be modified to prompt for this information, as the GetItems Groupware Connector
does. If the login is accepted by the Groupware Provider, the Connector prompts
the user to choose one of two actions: either retrieving the first message in
the user's specified folder, or adding a new item in the user's specified
folder.

The location of the custom items folder must be specified in the connector.ini
file. For accessing an Exchange store, only the Folder value need be specified.
To access a Domino store, both the Folder and the Database values must be
specified, since Domino uses a database/folder scheme to store data. To specify
a Domino database, use the file name (which ends with ".nsf") for the database,
not the Notes name.

This example can easily be modified to retrieve several items at once, or to
perform other Groupware actions (such as moving, updating and deleting). See
the other Groupware samples for more information on how to interact with
ThinAir Groupware Providers.

Note: This sample Connector is written for WML devices ONLY.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

    * groupware.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    CustomItemConnector.class - compiled Java code

    /src - java source files



-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  The included
MAKE script will compile the sample using the JDK, if available.

Install the compiled sample code and connector.ini configuration file into a
```

```
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the CustomItems Connector has been
loaded and initialized.  From a WML device, enter the IP address listed as the
value for ApplicationPath in connector.ini (your ThinAirServer IP address),
followed by /samples/custom.  For a machine with IP address 111.222.12.34 this
would be:

     http://111.222.12.34/samples/custom

Follow the on-screen instructions.

===============================================================================
                          Last updated: 11.17.2000

                     Copyright 1999, 2000 ThinAirApps Inc.
===============================================================================
```

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.provider.*;

//rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

// the groupware packages
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.* ;

import java.util.*;
import java.io.*;

/**This sample illustrates the use of the CustomItem type to handle data in custom-created
 * folders and databases.
 * This sample renders WML. It prompts the user to choose one of two actions: add (create a ↙
   new
 * item in the specified folder), or read (get the field names and values in the first item ↙
   found
 * within that folder, and display a group of them on the screen).
 *
 * The login data (provider name, host name, username, password), the name of the template/ ↙
   form for
 * the custom item folder, the name of the folder and (for a Lotus Domino item) the name of ↙
   the
 * database, are all specified within the connector.ini file.
 *
 * For a comprehensive reference of the Groupware Library see the ThinAir Groupware javadocs.
 */
public class CustomItemConnector implements Connector
{

     // The friendly name of this sample app
     protected String          appName;

     // Our access point to the services of ThinAir Server
     protected ConnectorAccess connectorAccess;

     // The application log
     protected com.thinairapps.platform.connector.ApplicationLog log;

     // The provider
     protected String provider;

     // The user's login data
     protected String host, userName, password;

     // The location of the custom item folder within the Groupware store.
     // This should not be a global variable in a real connector.
     protected String location;

     // The name of the form/template that this custom folder uses -
     // this variable is not actually used in any of this connector's code; however,
     // it was included because it would be used by any real connector dealing with
     // CustomItems, to add new items. See the comments within the addCustomItem()
     // method for details
```

```
    protected String formName;

    protected String sessionId = null;

    private String ACTION_FIELD = "action";
    private String LOGIN_ACTION = "login";
    private String CREATE_ACTION = "add";
    private String READ_ACTION = "read";



    /**
     * init() is called by the ThinAirServer when the Connector is loaded.  It provides the ↙
       connector with
     * resources it needs to interact with the ThinAirServer.
     * For more information about the Connector interface, see the javadocs for the ThinAir ↙
       Server API
     *
     * @param applicationName is a String derived from connector.ini.
     * @param applicationPath is a String derived from connector.ini.  We don't need this for↙
       this sample.
     * @param connectorProps is a Properties list containing developer assigned ↙
       connector-specific properties.
     * @param connectorAccess is our access point to the services provided by ThinAir Server.
     */
    public void init(String applicationName, String applicationPath, Properties ↙
        connectorProps,
                    ConnectorAccess connectorAccess, com.thinairapps.platform.connector. ↙
                       ApplicationLog appLog) throws ConnectorInitException
    {

        this.appName = applicationName;
        this.connectorAccess = connectorAccess;
        log = appLog;

        // the two strings from connector.ini that we'll use to create the official
        // "location" string
        String folder, database;

        // get provider name, as well as all login data, location of the custom folder, and
        // the name of the form/template being used, from the properties list (connector.ini)
        // Make sure you set up all necessary information before running this example.
        provider = connectorProps.getProperty("Provider");

        if (provider.length() == 0) throw new ConnectorInitException("No Provider entry in ↙
            connector.ini");

        host = connectorProps.getProperty("Host");
        if (host.length() == 0) throw new ConnectorInitException("No Host entry in connector.↙
            ini");

        userName = connectorProps.getProperty("UserName");
        if (userName.length() == 0) throw new ConnectorInitException("No UserName entry in ↙
            connector.ini");

        password = connectorProps.getProperty("Password");
        if (password.length() == 0) throw new ConnectorInitException("No Password entry in ↙
            connector.ini");

        folder = connectorProps.getProperty("Folder");
        if (folder.length() == 0) throw new ConnectorInitException("No Folder entry in ↙
            connector.ini");

        database = connectorProps.getProperty("Database");
        // no exception thrown if user didn't include the name of the database - this may or ↙
            may
        // not be a necessity for the groupware store being accessed. In the case of the ↙
            groupware
        // providers that come with the ThinAir Server, the Domino provider requires one, ↙
```

```
            while the
        // Exchange provider doesn't


        // now, set the location string - if no database name was included, then location
        // will just be equal to the folder name
        if (database.length() == 0)
        {
            location = folder;
        }
        else
        {
            // a database name was included; since we have only a single String to represent
            // the location within the eventual data request, how do we get both the folder
            // and the database name into this one String? Thankfully, there's a utility in
            // the CustomItem class that takes care of it for us
            location = CustomItem.LocNameUtils.createLocationString(database, folder);
        }

        formName = connectorProps.getProperty("FormName");
        // no exception thrown if user didn't include the name of the folder's form/template;
        // a CustomItem connector can function without it, although not as well
    }


    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a        ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing   ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used   ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer  ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this       ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String deviceType = "TA_WAP";
        String[] deviceTypes = {deviceType};
        return deviceTypes;
    }


    /**The handle method implements the core logic of a Connector.  It takes an incoming      ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,     ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters    ↙
```

```
        from the device.
     * @param device a Device object created in the image of the actual device making this ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream result) throws ↙
        IOException
    {

        String resultString = null;

        //get the 'action' parameter from the request. This is an HTTP param we define to ↙
            determine what action
        //to take when we get a request.

        String action = props.getProperty(ACTION_FIELD);

        try
        {
            if (action == null)
            {
                // if this is the first hit (or any request for the main deck), build a
                // deck that lets the user specify which action to run
                resultString = renderStartScreen();
            }
            else if (action.equals(LOGIN_ACTION))
            {

                sessionId = loginUser(provider, host, userName, password);
                resultString = renderOptionMenu();

            }
            else
            {
                if (action.equals(CREATE_ACTION))
                {

                    addCustomItem(location, sessionId);
                    resultString = renderMessage("Item successfully added!");

                }
                else if (action.equals(READ_ACTION))
                {

                    CustomItem item = getCustomItem(location, sessionId);

                    //render the fields of this object
                    resultString = renderCustomItemFields(item);

                }

                // log off the user
                connectorAccess.getStoreProvider(sessionId).disconnectUser();
                // then delete the session
                connectorAccess.deleteSession(sessionId);
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            // Here, we employ a primitive solution of simply displaying the error message ↙
                and providing a link
            // back to the welcome screen; a larger app would handle each error separately ↙
                and navigate the
            // user accordingly
            resultString = renderMessage(e.getMessage());
        }

        // in this example we logged on, performed a simple action, then logged off again. A ↙
        more
     // complete app would hold the session open between requests, and cache the retrieved
     // StoreItems in the session cache, using this to feed item bodies out to the client
     // without going to the provider each time

     byte[] resultBytes = resultString.getBytes();
     result.write(resultBytes);
 }



 /**loginUser() logs in the user to a groupware store using the specified login data
  *
  * @param providerName the name of the provider being used to access the message store.
  * @param host the IP or server name of the message store.
  * @param userName the user name of the account being logged onto.
  * @param password the password for this user.
  * @return a providerSessionId if success; otherwise an error will be thrown.
  */
 protected String loginUser (String providerName, String host, String userName,
                             String password) throws Exception
 {

     String SID = null;

     try
     {
         // Create a new Session with the specified provider and returns a unique Session ↙
             ID.
         SID = connectorAccess.createProviderSession (providerName);

         // Get the providerProxy associated with the session we just created,
         // this is what is used to interact with the Provider
         StoreProviderProxy spLite =  connectorAccess.getStoreProvider (SID);

         // Create a StoreProviderLogin object, this defines the action the provider will ↙
             execute
         StoreProviderLogin login = new StoreProviderLogin (userName, password, host);

         // use the providerProxy to login. The provider returns the items it supports
         SupportedItems supports = spLite.connectUser (login);

         // check to make sure that this provider handles CustomItem objects
         boolean supportsCustItems = false;
         Enumeration supportedEnum = supports.getItems();
         while (supportedEnum.hasMoreElements()) {
             SupportedItem curItem = (SupportedItem)supportedEnum.nextElement();
             if (curItem.getType() == ItemTypes.CUSTOM_ITEM)
                 supportsCustItems = true;
         }

         // if it doesn't handle CustomItem objects, throw an exception
         if (!supportsCustItems)
             throw new Exception("Specified provider (" + providerName + ") does not      ↙
                 support CustomItem handling");

     }
     catch (NoSuchProviderException e)
     {
         throw new Exception("No Provider named " + providerName + " was loaded by the    ↙
             ThinAir Server");
     }

     return SID;
 }


 /**getCustomItem() retrieves the first item from a groupware location, and returns a
```

```
     * CustomItem object containing all its information
     *
     * @param location The location in the groupware store being accessed
     * @param SID The session ID for the user's connection to the groupware store
     * @return a CustomItem representing the first item in the folder
     */
    protected CustomItem getCustomItem (String location, String SID) throws Exception
    {
        String resultString;

        ItemRequest iReq = new ItemRequest ();
        iReq.itemType = ItemTypes.CUSTOM_ITEM;
        iReq.itemLocation = location;
        iReq.max = 1;
        iReq.startID = null;
        iReq.bounds = null;

        UserDataRequest udReq = new UserDataRequest ();
        udReq.requests = new ItemRequest[1];
        udReq.requests[0] = iReq;

        UserData uData = connectorAccess.getStoreProvider(SID).getUserData(udReq);
        ItemRequestResponse irr = uData.responses[0];
        StoreItems customItems = irr.items;

        // we got back customItems; get the first element out (which is all
        // we requested)
        return (CustomItem)customItems.elementAt(0);

    }

    /**addCustomItem() adds an item to a groupware location containing custom-
     * definted items
     *
     * @param location The location in the groupware store being accessed
     * @param SID The session ID for the user's connection to the groupware store
     */
    protected void addCustomItem (String location, String SID) throws Exception
    {
        // In order to add a new item and populate its fields, we have to know
        // what the names of its fields are. If this were a real application,
        // we'd already know ahead of time what all the fields are named, and could
        // thus prompt the user for the values of those fields we wanted to populate.
        // The code would look something like:

        // CustomItem custItem = new CustomItem(formName);
        // custItem.addField(importantField1, importantField1UserValue);
        // custItem.addField(importantField2, importantField2UserValue);
        //  ...etc.

        // However, because this is a generic sample, we don't know what any
        // of the fields will be ahead of time. The following code, thus, is a
        // hack: we create a CustomItem object that has the properties of the
        // first item in this folder, using the getCustomItem() method. This
        // object now has all the fields that the items in the folder being
        // accessed contain (or at least all but the standard item fields -
        // see the discussion below). We then go through each of the fields and
        // set them to a new value, depending on their type. We then create
        // the item.

        CustomItem custItem = getCustomItem(location, SID);

        // let's loop through the fields in this new item and set some values

        // get an enumeration of all the custom fields
        Enumeration fieldEnum = custItem.getCustomFieldData().getFields();

        while (fieldEnum.hasMoreElements())
        {
            //get the next field
            Field thisField = (Field)fieldEnum.nextElement();

            if (thisField.getName().length() > 0) {

                String fieldName = thisField.getName();

                //check the type
                if (thisField.getType() == Field.BOOLEAN_VAL)
                {
                    //set all booleans to true
                    thisField.set(true);
                }
                else if (thisField.getType() == Field.DOUBLE_VAL)
                {
                    //set all doubles to be 123.123
                    thisField.set(123.123);
                }
                else if (thisField.getType() == Field.INT_VAL)
                {
                    //set all ints to 123
                    thisField.set(123);
                }
                else if (thisField.getType() == Field.LONG_VAL)
                {
                    //set all longs (this will include currency values) to 123.45
                    thisField.set(123.45);
                }
                else if (thisField.getType() == Field.STRING_VAL)
                {
                    thisField.set("New String!"); //set all strings to "New String!"
                }
                else if (thisField.getType() == Field.DATE_VAL)
                {
                    //set all dates to current time
                    thisField.set(new Date(System.currentTimeMillis()));
                }
            }
        }


        // That took care of all the custom fields. It may not have, however, taken care
        // of all the standard item fields, those fields that were present in the groupware
        // store template that this folder's form/template was derived from (if, in fact, it
        // was derived from another template). If we wanted to access these standard fields,
        // we would do:

        // StoreItem standardItem = item.getStandardItem();

        // Then you could treat standardItem like any other groupware StoreItem;
        // see the other groupware connector samples for more on how to manipulate standard ↙
            items

        // set the location of the new item to be the user-specified location
        custItem.setLocationInStore(location);

        StoreProviderProxy spProxy = connectorAccess.getStoreProvider (SID);

        AddNewGroupwareItem addAction = new AddNewGroupwareItem(custItem);

        spProxy.doUserDataAction (addAction);
        return;
    }


    /**This method renders a deck containing a welcome card
     *
     * @return the rendered deck.
     */
    private String renderStartScreen()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create a card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_WRAP);

        p.addChild(new Text("Custom Items"));
        p.addChild(new Break());
        p.addChild(new Text("Sample Connector"));

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_WRAP);

        // links to the two possible actions
        String loginHref = "?" + ACTION_FIELD + "=" + LOGIN_ACTION + "&amp;rnd=" + Math.
            random();

        // Go task for the href
        Go loginGo = new Go(loginHref,true,Go.METHOD_GET);

        // Anchor for the Go task
        Anchor loginAnchor = new Anchor(loginGo,new Text("Login"));

        //add the anchors to the Paragraph
        p.addChild(loginAnchor);

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the card to the deck
        deck.addCard(card1);

        String resultString = deck.render();

        return resultString;
    }


    /**This method renders a deck with a card that lets the user specify which action to take
     *
     * @return the rendered deck.
     */
    private String renderOptionMenu()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create a card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");
        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_WRAP);

        p.addChild(new Text("Custom Items"));
        p.addChild(new Break());
        p.addChild(new Text("Sample Connector"));

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_WRAP);
        // links to the two possible actions
        String createHref = "?" + ACTION_FIELD + "=" + CREATE_ACTION + "&amp;rnd=" + Math. ↙
            random();
        String readHref = "?" + ACTION_FIELD + "=" + READ_ACTION + "&amp;rnd=" + Math.random ↙
            ();

        // Go tasks for the two hrefs
        Go createGo = new Go(createHref,true,Go.METHOD_GET);
        Go readGo = new Go(readHref,true,Go.METHOD_GET);

        // Anchors for the two Go tasks
        Anchor createAnchor = new Anchor(createGo,new Text("Create a new item"));
        Anchor readAnchor = new Anchor(readGo,new Text("Read first item"));

        //add the anchors to the Paragraph
        p.addChild(createAnchor);
        p.addChild(readAnchor);

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the card to the deck
        deck.addCard(card1);

        String resultString = deck.render();

        return resultString;
    }



    /**This method renders a deck with one card containing the first 15 fields in
     * the CustomItem.
     *
     * @param item the CustomItem whose fields we should render
     * @return the rendered deck.
     */
    public String renderCustomItemFields (CustomItem item)
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();
        String url = null;

        //create the first card in the deck and give it the ID 'c1'
        DisplayCard card = new DisplayCard("c1");

        Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        p.addChild(new Text ("Custom Item"));

        p.addChild(new Break());
        card.addParagraph (p);

        Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        // Now add the fields and their values

        //first get the Data object that conatins all the info about our custom fields.
        Data customFields = item.getCustomFieldData();

        //we can get an enumeration of the fields...
        Enumeration fieldEnum = customFields.getFields();

        // go through the fields, and add each one to the deck - we'll stop after 15,
        // to avoid any deck overflow problems
        int itemsDisplayed = 0;
        while (fieldEnum.hasMoreElements() && itemsDisplayed < 15)
        {
            String fieldText = null;
            Field thisField = (Field)fieldEnum.nextElement();

            //you must check the type
            if (thisField.getType() == Field.BOOLEAN_VAL)
            {
                fieldText = thisField.getName()+": " + thisField.getBoolean();
            }
            else if (thisField.getType() == Field.DOUBLE_VAL)
            {
                fieldText = thisField.getName()+": " + thisField.getDouble();
            }
            else if (thisField.getType() == Field.INT_VAL)
            {
                fieldText = thisField.getName()+": " + thisField.getInt();
            }
            else if (thisField.getType() == Field.LONG_VAL)
            {
                fieldText = thisField.getName()+": " + thisField.getLong();
            }
            else if (thisField.getType() == Field.STRING_VAL)
            {
                fieldText = thisField.getName()+": " + thisField.getString();
            }
            else if (thisField.getType() == Field.DATE_VAL)
            {
                fieldText = thisField.getName()+": " + thisField.getDate();
            }

            p2.addChild(new Text(fieldText));
            p2.addChild(new Break());
            itemsDisplayed++;
        }

        // link home.
        String href = "?rnd="+Math.random();
        Go go = new Go(href,true,Go.METHOD_GET);
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        p2.addChild(anchor);

        card.addParagraph(p2);
        deck.addCard (card);

        return deck.render();
    }


    /** This method renders a simple message, either an error or a success,
     * then links back to the main page
     * @param message the message to be presented to the user
     * @return the rendered WML deck
     */
    private String renderMessage (String message)
    {
        WMLTagDocument deck = new WMLTagDocument();
        DisplayCard card = new DisplayCard();
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());


        String href = "?rnd="+Math.random();
        Go go = new Go(href,true,Go.METHOD_GET);
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        p.addChild(anchor);
        card.addParagraph(p);
        deck.addCard(card);

        String resultString = deck.render();

        return resultString;
    }
}
```

```
==============================================================================

                     WML Rendering Sample Connector

                     Wireless SDK for ThinAir Server

==============================================================================


-----------------
About this Sample
-----------------

This sample Connector demonstrates the use of the WML Tag library to accept
input from WML forms and render output for WML Browsers.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file.

    WMLSamplesConnector.class - compiled Java code

    /src - the Java source file, WMLSampleConnector.java



-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  Make sure to
append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the WML Rendering Connector has
been loaded and initialized. From your WML device, enter the IP address listed
as the value for ApplicationPath in connector.ini (your ThinAirServer IP
address), followed by /samples/wml.  For a machine with IP address
111.222.12.34 this would be:

     http://111.222.12.34/samples/wml

Follow the on-screen instructions.

==============================================================================

                     Last updated: 11.13.2000

              Copyright 1999, 2000 ThinAirApps Inc.
```

==

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;

//Rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

//Core Java API
import java.util.*;
import java.io.*;



/**
 * This is a simple sample whose purpose is to illustrate the use of the ThinAir WML Tag
 * Library.  It creates two simple WML decks: the first prompts the user for their favorite ↙
    color,
 * username and password, the second simply echos the submitted values. This sample makes
 * use of SelectInputCard, MultipleInputCard, DisplayCard, Do, Go, Anchor, Paragraph, Text, ↙
    and Break.
 *
 * For more information on use of the WML Tag Libraries, see the Tag Library documentation ↙
    and the ThinAir
 * Server Development Guide.
 */
public class WMLSampleConnector implements Connector
{
    //Declare variables global to this Connector
    String          appName;
    ConnectorAccess access;
    String          appPath;



    /**
     * init() is called by the ThinAirServer when the Connector is loaded.  It provides the ↙
        Connector with
     * resources it needs to interact with the ThinAirServer.
     *
     * @param applicationName is a String derived from connector.ini.  We don't need this for↙
        this sample.
     * @param applicationPath is a String dervid from connector.ini.  We don't need this for ↙
        this sample.
     * @param connectorProps is a Properties list containing developer assigned              ↙
        connector-specific properties.
     * We don't need this parameter in this sample.
     * @param connectorAccess is our access point to the services provided by ThinAir Server ↙
        .  We don't need this for this sample.
     * @param applicationLog is used for Logging.  We do not use this in this sample
     */
    public void init(String applicationName, String applicationPath, Properties              ↙
        connectorProps, ConnectorAccess connectorAccess, ApplicationLog applicationLog)
    {
        appName = applicationName;
        access = connectorAccess;
        appPath = applicationPath;
    }



    /**
     * getDevices() is called once by the ThinAir Server during start-up.  It allows a        ↙
```

```
      Connector to
    * indicate the types of devices it supports.  getDevices() returns an array containing ↙
      the names of all
    * DeviceProfiles supported by this Connector.  These names are the friendly names used ↙
      to uniquely
    * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
      to the ThinAir
    * Server Developer Guide or call DeviceProfile's getName() method.
    *
    * For more details about device detection and handling see the DeviceDetective sample ↙
      connector and the
    * ThinAir Server Developer Guide.
    *
    * @return an array of Strings representing the friendly names of the devices this ↙
      Connector supports.
    */
   public String[] getDevices()
   {
       String deviceType = "TA_WAP";
       String[] deviceTypes = {deviceType};
       return deviceTypes;
   }


   /**
    * The handle method implements the core logic of a Connector.  It takes an incoming ↙
      request from a
    * particular device, and returns an appropriate response. This method is called whenever↙
      the server
    * receives a request from a type of device that the Connector indicates it supports, ↙
      destined (as
    * indicated in the request URL) for a specific application. It is the responsibility of ↙
      the Connector
    * to interpret the request and generate an appropriate response.
    *
    * The server will pass a Device object containing as much information as possible into ↙
      this method.
    * The Connector can then utilize the particular Device class to determine more detailed ↙
      information
    * on the capabilities of the particular device making the request.
    *
    * @param props a set of name value pairs corresponding to the HTTP request parameters ↙
      from the device.
    * @param device a Device object created in the image of the actual device making this ↙
      request.
    * @param result a reference to the OutputStream that will be returned to the device.
    */
   public void handle(Properties props, Device device, OutputStream result)
   {
       String resultString = null;

       //Get the 'action' parameter from the request.
       //This is an HTTP param we define to determine what action to take when we get a ↙
           request.
       String action = props.getProperty("action");

       //If this is the first hit
       if (action == null)
       {
           //Build a deck that lets the user enter information.
           resultString = renderWelcome();
       }
       //If they have already entered the information, then display it
       else if (action.equals("display"))
       {
           //Build a display deck with the entered info, pass the request properties in
           resultString = renderAnswers(props);
       }
```

```
        byte[] resultBytes = resultString.getBytes();
        try
        {
            result.write(resultBytes);
        }
        catch (IOException e)
        {
            System.err.println("Error! Unable to write to result OutputStream.");
        }
    }


    /**
     * This method renders a deck with several cards including a welcome card and card for ↙
       entering information
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For ↙
       more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
       Development Guide.
     *
     * @return the rendered deck.
     */
    private String renderWelcome()
    {
        //Create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create the first card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //Create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_WRAP);

        p.addChild(new Text("Render WML"));
        p.addChild(new Break());
        p.addChild(new Text("Welcome"));
        p.addChild(new Break());

        //Add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_WRAP);

        //A link to the second card
        String href = appPath + "?#c2";

        //The go element is a task element that instructs the device to open a specified URL.
        Go go = new Go(href,true,Go.METHOD_GET);

        //The anchor element anchors a task to a string of formatted text. This is often ↙
            called a link.
        Anchor anchor = new Anchor(go,new Text("Next"));

        //Add the anchor to the Paragraph
        p.addChild(anchor);

        //Add the second Paragraph to the card
        card1.addParagraph(p);

        //Add the first card to the deck
        deck.addCard(card1);

        //Create a second card, give it the ID 'c2'
        SelectInputCard card2 = new SelectInputCard("c2");

        //SelectInputCards gives the user a choice list
        //Create all the choices and make "OK" the button label for all of these
        //A single letter (i.e. 'r', 'o', 'y' etc) will be the value assigned to
        //the variable in order to keep the size of the transmission down
        Option red = new Option("OK","r","Red");
        Option orange = new Option("OK","o","Orange");
        Option yellow = new Option("OK","y","Yellow");
        Option green = new Option ("OK","g","Green");
        Option blue = new Option ("OK","b","Blue");
        Option indigo = new Option("OK","i","Indigo");
        Option plaid = new Option("OK","p","Plaid");

        //Make an array of all the options
        Option[] options = {red,orange, yellow, green, blue, indigo, plaid};

        //Build the card.  Link it to card 3 (c3), set the prompt to be "Favorite color;",  ↙
            set
        //the variable name for this choice to be "color", then set the allignment and     ↙
            wrapping
        card2.buildCard("#c3","Favorite color:","color",options,Paragraph.ALIGN_LEFT,      ↙
            Paragraph.MODE_NOWRAP);

        //Add the SelectInputCard
        deck.addCard(card2);

        //Create a new MultipleInputCard and give it the ID 'c3'
        MultipleInputCard card3 = new MultipleInputCard("c3");

        //This allows the user to type in information
        LabeledInput userName = new LabeledInput("usr","Username:");

        //This sets the input text to lowecase by default though the user can change it
        userName.setFormat("*m");

        LabeledInput password = new LabeledInput("pwd","Password:");
        password.setFormat("*m");

        //This will display input characters as stars
        password.setType(Input.TYPE_PASSWORD);

        LabeledInput[] inputs = {userName,password};

        //Set the URL params to the values in the WML variables
        //&amp;, the escape sequence for ampersand, delimits name-
        //value pairs.  $ is used to dereference a WML variable.
        href = appPath + "?action=display&amp;color=$color&amp;usr=$usr&amp;pwd=$pwd&amp;rnd ↙
            ="+Math.random();

        //Build the card with the href, "Submit" as the button label, the array of Inputs,  ↙
            and the method specified.
        card3.buildCard(href,"Submit",inputs, Go.METHOD_GET);
        deck.addCard(card3);

        //Render the deck
        return deck.render();
    }


    private String renderAnswers(Properties props)
    {
        //Get the arguments passed from the welcome deck
        String usrName = props.getProperty("usr");
        String password = props.getProperty("pwd");

        //This will be a single letter, so we have to map it to a color
        String color = props.getProperty("color");
        if (color == null)
            return renderException("Error! No color was entered.");
        if (color.equals("r"))
```

```
            color = "Red";
        else if (color.equals("o"))
            color = "Orange";
        else if (color.equals("y"))
            color = "Yellow";
        else if (color.equals("g"))
            color = "Green";
        else if (color.equals("b"))
            color = "Blue";
        else if (color.equals("i"))
            color = "Indigo";
        else if (color.equals("p"))
            color = "Plaid";

        //Create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create the first card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //Create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_WRAP);

        //Display the values
        p.addChild(new Text(color));
        p.addChild(new Break());
        p.addChild(new Text(usrName));
        p.addChild(new Break());

        //Show the password for this sample.  In general, of course, this is not recommended ↙
            ...
        p.addChild(new Text(password));

        card1.addParagraph(p);

        //We add a random number to prevent unwanted caching by some browsers
        String href = appPath + "?rnd="+Math.random();

        //The go element is a task element that instructs the device to open a specified URL.
        //We pass an href with no action, this will bring us to the welcome page
        Go go = new Go(href,true);

        //This will make a button with the label 'Main'
        Do dew = new Do(Do.TYPE_ACCEPT,go,"Main","main",false);

        //Add the button to the card
        card1.addChild(dew);

        //Add the card to the deck
        deck.addCard(card1);

        //Render the deck
        return deck.render();
    }


    /**
     * This is a simple exception rendering method.
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For    ↙
       more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
       Development Guide.
     *
     * @param message the message to be presented to the user
     * @return the rendered WML deck
     */
    private String renderException (String message)
    {
        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a display card
        DisplayCard card = new DisplayCard();

        //Create a new paragraph
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());

        //Create the URL
        String href = appPath + "?rnd="+Math.random();

        //The go element is a task element that instructs the device to open a specified URL.
        Go go = new Go(href,true,Go.METHOD_GET);

        //The anchor element anchors a task to a string of formatted text. This is often     ↙
            called a link.
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        //Add the anchor to the paragraph
        p.addChild(anchor);

        //Add the paragraph to the card
        card.addParagraph(p);

        //Add the card to the deck
        deck.addCard(card);

        //Render the deck
        return deck.render();
    }
}
```

```
==============================================================================

                     Profile Management Sample Connector

                       Wireless SDK for ThinAir Server

==============================================================================


-----------------
About this Sample
-----------------

This sample Connector demonstrates how to take advantage of ThinAir Server's
Profile Management features.  In the ThinAir Server architecture, User Profiles
are server-wide, password-protected records that are available to all
Connectors.  A Connector can store application-specific data within the User
Profile, and query the profile for data such as the user's known Devices.

In this simple example the Connector prompts the user for a number and stores
it in their User Profile as application data.  The Connector then displays this
information to the user.

This Connector also makes use of Session objects.  For more information on
using
Sessions, see the SessionManagement sample Connector in this directory and the
corresponding ThinAir Server API documentation.

N.B. This sample Connector is written for WML devices ONLY.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    ProfileConnector.jar - compiled Java code

    /src - java source files - ProfileConnector.java and ProfileData.java


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  Make sure to
append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the Profile Management Connector
has been loaded and initialized. From your WML device, enter the IP address
listed as the value for ApplicationPath in connector.ini  (your ThinAirServer
IP address), followed by /samples/profile. For a machine with IP address
```

111.222.12.34 this would be:

    http://111.222.12.34/samples/profile

Follow the on-screen instructions.

==

Last updated: 11.13.2000

==

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//This is a simple class which is the container for user profile data
import java.io.Serializable;

public class ProfileData implements Serializable
    {
        public String favoriteNumber;
    }
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.exception.*;

//Rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

//Core Java API
import java.util.*;
import java.io.*;



/**This example illustrates how to user ThinAir Server's Profile management features.
 * Because Profiles are usually most effective and useful in stateful applications,
 * this example also makes use of sessions as well. For more information on using
 * sessions, see the SessionConnector sample, the ThinAir Server API documentation,
 * and the ThinAir Server Development Guide.
 *
 * The ThinAir Server creates a password-protected User Profile for each user of the
 * system so the user must first log in.  This User Profile is identified by a globally
 * unique ID (the ThinAir User ID).  User Profiles include information about which
 * devices a user owns, which applications have been accessed by users and any application
 * specific data (such as the back-end server account information).  ThinAir Server
 * administrators have control over all User Profiles through the User Manager tool
 * that allows them to view all Profiles and add or remove users from both individual
 * applications and the ThinAir Server itself.
 *
 * The Profile Connector follows a standard design pattern for ThinAir Connectors that
 * require user profiles. Once users log in, they are then shown a menu with various
 * application features presented as options.  In this simple example, users can set a
 * favorite number and store it in their profile.  In subsequent screens, they can then
 * view it.  Returning users are handled in one of two ways depending on whether they have
 * a device GUID.  If the user has a device GUID, then the application looks up their User
 * Profile based on that GUID and they don't have to log in each time that they access the
 * application.  Returning users without a device GUID are forced to enter in their User ID
 * (which corresponds directly to their UserProfileID internally) in order for the
   application
 * to recognize them (i.e. in order for them to have access to their User Profile).
 */
public class ProfileConnector implements Connector
{
    //Declare variables global to this Connector
    String          appName;
    ConnectorAccess access;
    String          appPath;

    //We'll use this as the param name for user profiles
    public final static String USER_PROFILE_ID_ARG = "userProfileID";

    //This will be the param name for passwords
    public final static String USER_PROFILE_PWD_ARG = "pw";

    //The param name for the favorite number
    public final static String FAVORITE_NUMBER = "num";



    /**init() is called by the ThinAirServer when the Connector is loaded.  It provides the
     * Connector with resources it needs to interact with the ThinAirServer.
     *
     * @param applicationName indicates the friendly name of this Connector application.
     *                        It is a String derived from connector.ini and this sample does
     *                        utilize it.
     * @param applicationPath is the path to this Connector application.
     *                        It is a String derived from connector.ini and this sample does
     *                        utilize it.
     * @param connectorProps is a Properties object containing developer assigned,         ↙
       connector-specific
     *                        properties. It is derived from connector.ini and this sample  ↙
       does not
     *                        utilize it.
     * @param connectorAccess is the one-and-only interface a Connector obtains to gain     ↙
       access to
     *                        the runtime services (such as session and user profile        ↙
       management)
     *                        offered by the ThinAir Server to running Connectors.  This    ↙
       sample uses
     *                        it to create a profile and store and retrieve data from the   ↙
       session cache.
     *
     * @param appLog is used for Logging
     */
    public void init(String applicationName, String applicationPath, Properties            ↙
        connectorProps, ConnectorAccess connectorAccess, com.thinairapps.platform.connector. ↙
        ApplicationLog appLog)
    {
        appName = applicationName;
        access = connectorAccess;
        appPath = applicationPath;
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a      ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing ↙
        the
     * names of all DeviceProfiles supported by this Connector.  These names are the friendly↙
        names
     * used to uniquely identify every DeviceProfile.  To get the friendly name of a         ↙
        particular device,
     * refer to the ThinAir Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
        Connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this      ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String deviceType = "TA_WAP";
        String[] deviceTypes = {deviceType};
        return deviceTypes;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming     ↙
        request
     * from a particular device, and returns an appropriate response. This method is called  ↙
        whenever
     * the server receives a request from a type of device that the Connector indicates it
     * supports, destined (as indicated in the request URL) for a specific application. It is
     * the responsibility of the Connector to interpret the request and generate an          ↙
        appropriate
     * response.
     *
     * The server will pass a Device object containing as much information as possible into ↙
       this
     * method. The Connector can then utilize the particular Device class to determine more ↙
       detailed
     * information on the capabilities of the particular device making the request.
     *
     * @param props is a set of name value pairs corresponding to the HTTP request parameters↙
       from
     *              the device.
     * @param device is a Device object created in the image of the actual device making this↙
       request.
     * @param result is a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream result)
    {
        //Within the renderLogin method, we name the sessionID paramater in the URL "sid".
        //We get the value here.
        String sessionID = props.getProperty("sid");

        //Find out what action the user is trying to perform
        String action = props.getProperty("action");

       //The User Profile is identified by a globally unique ID (the user's ThinAir User ID)
        String userProfileID = null;

        //The page to be rendered
        String resultString = null;

        //The cache for this session
        Hashtable cache = null;

        //If this is the user's first hit, they will not yet have a session
        if (sessionID == null)
        {
            //So create one for them
            sessionID = access.createSession();
        }

        //ThinAir Server administrators have control (through the User Manager tool) over    ↙
            whether
        //password authentication is required for each application running on the ThinAir   ↙
            Server.
        boolean mustAuthenticate = true;

        //Here we check if the session needs to be authenticated
        try
        {
            mustAuthenticate = access.userAuthenticationRequiredForSession(appName,          ↙
                sessionID);
        }
        //Handle the case where the session has timed out
        catch (NoSuchSessionException e)
        {
            renderException("Your session has timed out.  Please re-register.");
        }

        //If the ThinAir Server administrator has not required password authentication, then ↙
            he or she
        //is allowing known devices to logon automatically.
        if (!mustAuthenticate)
        {
            //Get the device GUID.  getGuid() returns null for those devices that don't have ↙
                GUIDs
            String deviceGUID = device.getGUID();

            //If the device has a deviceGUID and the userProfileID still hasn't been assigned
            if (userProfileID == null && deviceGUID != null)
            {
                //Look up a User Profile ID from a Device GUID
            //This userProfileID will later be used to both set and retrieve User Profile↙
               information
            userProfileID = access.getUPIDFromDeviceGUID(deviceGUID);
        }
    }

    //If the user has not performed an action and userProfileID is still null, then they ↙
        need to login
    if (action == null && userProfileID == null)
    {
        resultString = renderLogin(sessionID);
    }
    //If we have their userProfileID from their deviceGUID and there is no action        ↙
        performed by the user
    else if (action == null && userProfileID != null)
    {
        try
        {
            //Set the session as authenticated
            access.setSessionAuthenticated(sessionID, true);

            //Gets the caller-modifiable cache for a specified session.
            cache = access.getSessionCache(sessionID);

            //Place the userProfileID in the cache
            cache.put(USER_PROFILE_ID_ARG, userProfileID);

            //Render the menu
            resultString = renderMenu(sessionID);
        }
        //Handle an expired session
        catch (NoSuchSessionException e)
        {
            resultString = renderException("Your session has timed out.  Please          ↙
                re-register.");
        }
    }
    //Display the menu
    else if (action != null && action.equals("main"))
    {
        resultString = renderMenu(sessionID);
    }
    //Change the profile data
    else if (action != null && action.equals("update"))
    {
        resultString = updateProfileData(props, sessionID);
    }
    //Show the profile data
    else if (action != null && action.equals("show"))
    {
        resultString = renderShowNumber(sessionID);
    }
    //Log in the user
    else if (action != null && action.equals("login"))
    {
        resultString = login(props, device);
    }
    //Show the data entry deck
    else if (action != null && action.equals("set"))
    {

        resultString = this.renderSetNumber(sessionID);
    }

    //Render for the device by turning the String into an array of bytes
    byte resultBytes[] = resultString.getBytes();
    try
    {
        //Write the bytes to the output stream
```

```
            result.write(resultBytes);
        }
        catch (IOException e)
        {
            System.err.println("Error! cannot write to result OutputStream.");
        }
    }



    /**
     * This method renders a deck to allow the user to enter a User ID and password.
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For ↙
       more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
       Development Guide.
     *
     * @param sessionID the unique sessionID for the user.
     * @return A String containing the WML deck to be display to the user.
     */
    private String renderLogin(String sessionID)
    {
        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //The data will be sent to the server by means of a POST so no $variables need to be ↙
            set in the URL

        //Build the URL...
        //Some devices cache content more than they should. Adding a random parameter is an ↙
            unbeautiful,
        //though often necessary technique for tricking the phone into always hitting the ↙
            server rather
        //than getting a page from its local cache.
        //n.b. Certain phones (such as the Nokia WAP Toolkit Version 2.0 simulator) require ↙
            the absolute
        //application path so we include it here
        String url = appPath+ "?action=login&amp;sid="+ sessionID + "&amp;rnd=" + Math.random↙
            ();

        //Create a display card
        Card card = new Card("u1","Welcome");

        //The Go element is a task element that instructs the device to open a specified URL.
        Go go = new Go (url,true,Go.METHOD_POST);

        LabeledInput userName = new LabeledInput(USER_PROFILE_ID_ARG,"text","*m","User ID:");
        LabeledInput password = new LabeledInput(USER_PROFILE_PWD_ARG, "password","* ↙
            m","Password:");
        LabeledInput[] inputs = {userName,password};

        //Add the post fields
        for (int i = 0; i < inputs.length; i++)
            go.addChild(new PostField(inputs[i].getInputName(),"$" + inputs[i].getInputName ↙
                ()));

        //Create a Do element that associates a task with an element within the user ↙
            interface.
        //When the user invokes the user interface mechanism, the device performs the ↙
            associated element task.
        Do dew = new Do(Do.TYPE_ACCEPT,go);

        dew.addAttribute("label","Next");

        //Add the Do element to the card
        card.addChild(dew);

        //Create a Paragraph
        Paragraph p = new Paragraph();

        //Add text to the paragraph
        p.addChild(new Text("Welcome to the Profile Connector"));
        p.addChild(new Break());

        for (int i = 0; i < inputs.length; i++)
            p.addChild(inputs[i]);

        //Add the paragraph to the card
        card.addChild(p);

        //Add the card to the deck
        deck.addChild(card);

        //Render the deck
        return deck.render();
    }


    /**
     * This generates the main menu deck.  The menu gives the user the option to either set ↙
       their
     * favorite number or view it.  This method is first called after login.
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For ↙
       more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
       Development Guide.
     *
     * @param sessionID - the unique session ID for the user.
     * @return A String containing the WML deck to be display to the user.
     */
    private String renderMenu(String sessionID)
    {
        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a display card
        DisplayCard card = new DisplayCard();

        //Create a Paragraph
        Paragraph p = new Paragraph();

        p.addChild(new Text("Menu"));
        p.addChild(new Break());
        p.addChild(new Break());

        //Build the URL...
        //Some devices cache content more than they should. Adding a random parameter is an ↙
            unbeautiful,
        //though often necessary technique for tricking the phone into always hitting the ↙
            server rather
        //than getting a page from its local cache.
        //n.b. Certain phones (such as the Nokia WAP Toolkit Version 2.0 simulator) require ↙
            the absolute
        //application path so we include it here
        String href = appPath+ "?action=set" + "&amp;sid=" + sessionID + "&amp;rnd=" + Math. ↙
            random();

        //The Go element is a task element that instructs the device to open a specified URL.
        Go go = new Go(href,true,Go.METHOD_GET);

        //The Anchor element anchors a task to a string of formatted text. This is often ↙
            called a link.
        Anchor a= new Anchor(go,new Text("Set Number"));
        p.addChild(a);
        p.addChild(new Break());
        //This is the action that will occur when the link is selected
        href = appPath+ "?action=show" + "&amp;sid=" + sessionID + "&amp;rnd=" + Math.random ↙
            ();

        //The second link
        go = new Go(href,true,Go.METHOD_GET);

        //Create the second link
        a= new Anchor(go,new Text("Show Number"));
        p.addChild(a);
        p.addChild(new Break());

        //Add the Paragraph to the card
        card.addParagraph(p);

        //Add the card to the deck
        deck.addCard(card);

        //Render the deck
        return deck.render();
    }


    /**
     * This method updates User Profile Data after the user has entered a new number.
     *
     * @param props - The Properties object containing the request parameters.
     * @param sessionID - The user's unique session ID
     */
    private String updateProfileData(Properties props, String sessionID)
    {
        Hashtable cache = null;
        String userProfileID = null;

        //Get the number from the URL
        String newNumber = props.getProperty(FAVORITE_NUMBER);

        //If the number is null, then fill it in
        if (newNumber == null)
            newNumber = "Not Set";
        try
        {
            //Get the session cache
            cache = access.getSessionCache(sessionID);

            //Get the userProfileID from the session cache
            userProfileID = (String)cache.get(USER_PROFILE_ID_ARG);

            //Get the old data from the profile
            ProfileData data = (ProfileData)access.getUserProfileData(userProfileID,appName);
            if (data != null)
            {
                //Set the value to the newly entered number
                data.favoriteNumber = newNumber;
            }
            else
            {
                //Create a new profile data container
                data = new ProfileData();

                //And fill it with our new number
                data.favoriteNumber = newNumber;
            }
            //Set the profile data to our profile data container object
            access.setUserProfileData(userProfileID,appName,data);

            //Bring the user directly to the display deck
            return renderShowNumber(sessionID);
        }
        //Handle the case where the UserManager has changed permissions on this app
        catch (AddAppDataPermissionException e)
        {
            return renderException("The Add Application Data Permission has been turned off ↙
                for this application");
        }
        //Handle the possibility of a time-out
        catch (NoSuchSessionException e)
        {
            return renderException("Your session has timed out. Please re-register.");
        }
        //Handle the possibility that the profile no longer exists.
        catch (NoSuchUserProfileException e)
        {
            return renderException("Your Profile has been deleted.  Please reregister or   ↙
                contact your administrator.");
        }
        //Handle the possibility that the administrator is using UserManager right now
        catch (ProfileStoreLockedException e)
        {
            return renderException("The profile store cannot be updated because the Profile ↙
                Store is locked.  Please try again or contact your administrator.");
        }
    }



    /**
     * renderShowNumber creates the markup document that displays the number.  It is either ↙
        called
     * when the user selects 'show number' from the main menu or after the user has set the ↙
        number.
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For    ↙
        more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
        Development Guide.
     *
     * @param sessionID the unique ID for this user's session.  This will be displayed to the↙
        user.
     * @return A String containing the WML deck to be display to the user.
     */
    private String renderShowNumber(String sessionID)
    {
        Hashtable cache = null;
        String userProfileID = null;
        String number = null;
        try
        {
            //Get the session cache
            cache = access.getSessionCache(sessionID);

            //Get the User Profile ID from the session cache
            userProfileID = (String)cache.get(USER_PROFILE_ID_ARG);

            ProfileData data = (ProfileData)access.getUserProfileData(userProfileID,appName);

            //The data has not yet been set
            if (data == null)
                number = "Not Set";
            else
                number = data.favoriteNumber;
        }
        catch (NoSuchSessionException e)
        {
            return renderException("Your session has timed out.  Please re-register.");
        }
        //Handle the rare case where the profile has been deleted by someone during our ↙
            session
        catch (NoSuchUserProfileException e)
        {
            return renderException("Your profile has been removed.  Please re-register or ↙
                contact your administrator.");
        }

        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a display card
        DisplayCard card = new DisplayCard();

        //Create a Paragraph
        Paragraph p = new Paragraph();

        //Display the user's sessionID
        p.addChild(new Text("Your number is: "+ number));

        //Build the URL...
        //Some devices cache content more than they should. Adding a random parameter is an ↙
            unbeautiful,
        //though often necessary technique for tricking the phone into always hitting the ↙
            server rather
        //than getting a page from its local cache.
        //n.b. Certain phones (such as the Nokia WAP Toolkit Version 2.0 simulator) require ↙
            the absolute
        //application path so we include it here
        String href = appPath+ "?action=main" + "&amp;sid=" + sessionID + "&amp;rnd=" + Math.↙
            random();

        //The go element is a task element that instructs the device to open a specified URL.
        Go go = new Go(href,true,Go.METHOD_GET);

        //Create a do element that associates a task with an element within the user ↙
            interface.
        //When the user invokes the user interface mechanism, the device performs the ↙
            associated element task.
        Do dew = new Do(Do.TYPE_ACCEPT,go);

        p.addChild(dew);

        //Add the Paragraph to the card
        card.addParagraph(p);

        //Add the card to the deck
        deck.addCard(card);

        //Render the deck
        return deck.render();
    }


    /**
     * This is a simple card that renders a user interface that allows the user to set a ↙
       number
     * that will be saved in the user's User Profile.
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For ↙
       more
     * information on use of the Tag Libraries, see the Tag Library documentation and the
     * ThinAir Server Development Guide.
     *
     * @param sessionID the uniques ID for the user's session.
     * @return A String containing the WML deck to be display to the user.
     */
    private String renderSetNumber(String sessionID)
    {
        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a display card
        MultipleInputCard card = new MultipleInputCard();

        //Create a Paragraph
        Paragraph p = new Paragraph();

        //set the action
        //Build the URL...
        //Some devices cache content more than they should. Adding a random parameter is an ↙
            unbeautiful,
        //though often necessary technique for tricking the phone into always hitting the   ↙
            server rather
        //than getting a page from its local cache.
        //n.b. Certain phones (such as the Nokia WAP Toolkit Version 2.0 simulator) require ↙
            the absolute
        //application path so we include it here
        String href = appPath+ "?action=update&amp;" + "sid=" + sessionID + "&amp;rnd=" +   ↙
            Math.random() + "&amp;"+ FAVORITE_NUMBER +"=$"+FAVORITE_NUMBER;

        //*N means that only numbers can be entered
        LabeledInput number = new LabeledInput(FAVORITE_NUMBER,"text","*N","Pick a number:");
        LabeledInput[] inputs = {number};
        card.buildCard(href,"submit",inputs,Go.METHOD_GET);

        //Add the card to the deck
        deck.addCard(card);

        //Render the deck
        return deck.render();
    }


    /**
     * The login method gets called after the user has entered their user ID and password. If↙
       the
     * userProfileID they entered already exists, it tries to authenticate the password. If  ↙
       the
     * password is wrong, it returns an error saying that this is the case.  If the password
     * is correct, it stores the userProfileID in the session cache for future use, sets
     * the session to be authenticated and renders the main menu.
     *
     * If the userProfileID the user entered does not exist, then the Connector tries to     ↙
       create one with the
     * given userProfileID.  This will work unless the administrator has set permissions to  ↙
       not allow the
     * creation of new user profiles.  If the Connector does not have permission, then a     ↙
       message
     * saying that will be returned.
     *
     * @param props The properties object passed into handle containing the request arguments
     * @param device the device making the request.  We use this when we create the profile  ↙
       so that it can be
     * associated with the user.
     */
    private String login(Properties props, Device device)
    {
        //Get the sessionID from the request
        String sessionID = props.getProperty("sid");

        //Get the user profile ID from the request
        String userProfileID = props.getProperty(USER_PROFILE_ID_ARG);

        //Get the password from the request
        String pass = props.getProperty(USER_PROFILE_PWD_ARG);
        Hashtable cache = null;

        //Check to see if the User Profile exists
        if (access.userProfileExists(userProfileID))
        {
            boolean valid = false;
            try
            {
                //Authenticate the User Profile ID and password
                valid = access.authenticateUser(userProfileID,pass);

                //If the login was successful
                if (valid)
                {
                    //Set the session to be authenticated
                    access.setSessionAuthenticated(sessionID,true);

                    //Get the session cache
                    cache = access.getSessionCache(sessionID);

                    //Place the userProfileID in the cache
                    cache.put(USER_PROFILE_ID_ARG,userProfileID);

                    //Bring them to the main page
                    return renderMenu(sessionID);
                }
            }
            //A rare case where the profile was deleted between our call to access.    ↙
                userProfileExists(userProfileID)
            //and access.authenticateUser(userProfileID,pass);
            catch (NoSuchUserProfileException e )
            {
                return renderException("Your profile has been deleted or changed. Please    ↙
                    re-register or consult your administartor.");
            }
            //Their session has timed out
            catch (NoSuchSessionException e)
            {
                return renderException("Your session has timed out. Please log in again.");
            }
        }
        //The userProfileID they entered is not in the profile store, so try to create it
        else
        {
            try
            {
                //This creates the profile and adds the device to its list
                access.createUserProfile(userProfileID,pass,appName,device);

                //Set the session as authenticated
                access.setSessionAuthenticated(sessionID,true);

                //Get the session cache
                cache = access.getSessionCache(sessionID);

                //Place the userProfileID in the cache
                cache.put(USER_PROFILE_ID_ARG,userProfileID);

                //Bring them to the main page.
                return renderMenu(sessionID);
            }
            //A rare case where someone else has created a User Profile with the same name
            catch (ProfileAlreadyExistsException e)
            {
                return renderException("This Profile ID is already taken.  Please choose    ↙
                    another.");
            }
            //If the administrator has set permission to disallow connectors from creating    ↙
```

```
                profiles
            catch(AddProfilePermissionException e)
            {
                return renderException("The ThinAir Server administrator has disallowed the ↙
                    creation of new profiles!");
            }
            //The administrator has locked the store to run UserManager
            catch(ProfileStoreLockedException e)
            {
                return renderException("The ThinAir Server profile store is currently being ↙
                    configured.  Try again later.");
            }
            //Their session has timed out
            catch (NoSuchSessionException e)
            {
                return renderException("Your session has timed out.  Please re-register.");
            }
        }
        //If none of the above exceptions occurred, then they entered an invalid ↙
            userProfileID
        //or a bad password or both
        return renderException("Your user ID or password was incorrect.  Please try again.");
    }


    /**
     * renderException creates a markup document with an error message.
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For ↙
       more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
       Development Guide.
     *
     * @param message The error message to be displayed to the user.
     * @return A String containing the WML deck to be display to the user.
     */
    private String renderException (String message)
    {
        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a display card
        DisplayCard card = new DisplayCard();

        //Create a new paragraph
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());

        //Create the URL
        String href = appPath+ "?rnd="+Math.random();

        //The go element is a task element that instructs the device to open a specified URL.
        Go go = new Go(href,true,Go.METHOD_GET);

        //The anchor element anchors a task to a string of formatted text. This is often ↙
            called a link.
        Anchor anchor = new Anchor(go,new Text("Start again..."));

        //Add the anchor to the the paragraph
        p.addChild(anchor);

        //Add the paragraph to the card
        card.addParagraph(p);

        //Add the card to the deck
        deck.addCard(card);
        //Returns the entire rendered document text, suitable for display in an WML browser
        return deck.render();
    }
}
```

```
==============================================================================

                        Session Management Sample Connector

                        Wireless SDK for ThinAir Server v1.2

==============================================================================


-----------------
About this Sample
-----------------

This sample Connector demonstrates the use of Sessions within the ThinAir
Connector API. When the user first contacts the server, the Session Management
Connector creates a Session object for that user and assigns it a unique
session identifier. This session ID is then passed along back and forth to
the device as a parameter of the HTTP request string.  In this way the session
persists each time the 'Hit again' button is pressed, while the 'Hit #'
increases.


N.B. This sample Connector is written for WML devices ONLY.


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar

This sample does not require any other external APIs.


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    SessionConnector.class - compiled Java code

    /src - java source file - SessionConnector.java


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  Make sure to
append the required class files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the Session Management Connector
has been loaded and initialized.  From your wireless HTML device, or web
browser, enter the IP address listed as the value for ApplicationPath in
connector.ini (your ThinAirServer IP address), followed by /samples/session.
For a machine with IP address 111.222.12.34 this would be:

     http://111.222.12.34/samples/session

Follow the on-screen instructions.
```

```
==============================================================================
                        Last updated: 11.13.2000
                  Copyright 1999, 2000 ThinAirApps Inc.
==============================================================================
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Core ThinAir Server API functionality
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.device.*;

//Rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

//Core Java API
import java.util.*;
import java.io.*;



/**
 * @(#)SessionConnector.java
 *
 * This sample Connector demonstrates the use of sessions within the ThinAir Connector API.
 * A session is created when the user first contacts the Connector.  The Connector generates
 * a unique session ID that is then passed back and forth between the server and client with
 * each HTTP request and response.  When using HTTP GET, this means adding a parameter to the
 * URL that specifies the session ID.  With HTTP POST, it involves adding a POST parameter   ↙
   for
 * the session ID in much the same way.
 */
public class SessionConnector implements Connector
{
    ConnectorAccess access;
    String          appPath;


    /**init() is called by the ThinAirServer when the Connector is loaded.  It provides the  ↙
        Connector with
     * resources it needs to interact with the ThinAirServer.
     *
     * @param applicationName indicates the friendly name of the application of which this  ↙
        Connector is a part.
     *                        It is a String derived from connector.ini and this sample does ↙
        not utilize it.
     * @param applicationPath is the path to the application of which this Connector is a   ↙
        part.
     *                        It is a String derived from connector.ini and this sample does ↙
        utilize it.
     * @param connectorProps is a Properties object containing developer assigned,          ↙
        connector-specific properties.
     *                        It is derived from connector.ini and this sample does not      ↙
        utilize it.
     * @param connectorAccess is the one-and-only interface a Connector obtains to gain      ↙
        access to the runtime
     *                        services offered by the ThinAir Server to running Connectors.  ↙
        This sample uses
     *                        it to create a session and store and retrieve data from the    ↙
        session cache.
     * @param appLog is used for Logging
     */
    public void init(String applicationName, String applicationPath, Properties              ↙
        connectorProps, ConnectorAccess connectorAccess, com.thinairapps.platform.connector. ↙
        ApplicationLog appLog)
    {
        access = connectorAccess;
        appPath = applicationPath;
    }
    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a     ↙
     Connector to
      * indicate the types of devices it supports.  getDevices() returns an array containing ↙
         the names of all
      * DeviceProfiles supported by this Connector.  These names are the friendly names used ↙
         to uniquely
      * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
         to the ThinAir
      * Server Developer Guide or call DeviceProfile's getName() method.
      *
      * For more details about device detection and handling see the DeviceDetective sample  ↙
         connector and the
      * ThinAir Server Developer Guide.
      *
      * @return an array of Strings representing the friendly names of the devices this      ↙
         Connector supports.
      */
     public String[] getDevices()
     {
         String deviceType = "TA_WAP";
         String[] deviceTypes = {deviceType};

         return deviceTypes;
     }



     /**The handle method implements the core logic of a Connector.  It takes an incoming   ↙
         request from a
      * particular device, and returns an appropriate response. This method is called whenever↙
         the server
      * receives a request from a type of device that the Connector indicates it supports,   ↙
         destined (as
      * indicated in the request URL) for a specific application. It is the responsibility of ↙
         the Connector
      * to interpret the request and generate an appropriate response.
      *
      * The server will pass a Device object containing as much information as possible into ↙
         this method.
      * The Connector can then utilize the particular Device class to determine more detailed ↙
         information
      * on the capabilities of the particular device making the request.
      *
      * @param reqProps - represents the HTTP request
      * @param device - the actual wireless device instance making the request
      * @param out - the OutputStream to write back the response
      */
     public void handle(Properties props, Device device, OutputStream result)
     {
         //A count of the number of times the user has hit the server during this session
         Integer hitNumber = null;
         //Within the renderPage method, we name the sessionID parameter in the URL "sid"
         String sessionID = props.getProperty("sid");
         //The cache for this session
         Hashtable cache = null;

         //If this is the user's first hit, they will not yet have a session
         if (sessionID == null)
         {
             //So create one for them
             sessionID = access.createSession();
         }
         try
         {
             //Get the cache for this session
             cache = access.getSessionCache(sessionID);
```

```
        }
        catch (NoSuchSessionException e)
        {
            //Can pass any exception messages to the exception rendering method
        }

        //Get a reference to the value to which the key is mapped in the cache hashtable
        hitNumber = (Integer)cache.get("hit");

        //The value is null if the key is not yet mapped to any value in the cache hashtable
        //This must be the user's first time through
        if (hitNumber == null)
        {
            //Create the first hit
            hitNumber = new Integer(1);

            //And store it in the cache
            cache.put("hit",hitNumber);
        }
        //They have been here before
        else
        {
            //So increment the existing count
            hitNumber = new Integer(hitNumber.intValue() + 1);

            //And store it in the cache
            cache.put("hit",hitNumber);
        }

        //Now render the result
        String resultString = renderPage (sessionID, hitNumber);

        //Turn the String into an array of bytes
        byte resultBytes[] = resultString.getBytes();

        try
        {
            //Write the bytes to the outputStream
            result.write(resultBytes);
        }
        catch (IOException e)
        {
            //Write to the 'standard' error output stream
            System.err.println("Error! cannot write to result OutputStream.");
        }
    }


    /**renderPage creates the markup document to be displayed.  This sample only supports WAP↙
        devices
     * (as indicated in the getDevices() method) and it makes use of the ThinAir WML Tag     ↙
        Library for
     * WML markup creation.
     *
     * @param sessionID is the unique ID for this user's session.  This will be displayed to ↙
        the user.
     * @param hitNumber is an Integer representing how many times the user has hit the server↙
        during this session.
     *
     * @return a String containing the WML deck to be display to the user.
     */
    private String renderPage(String sessionID, Integer hitNumber)
    {
        //Create a WML deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a display card
        DisplayCard card = new DisplayCard();
        //Create a Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_RIGHT, Paragraph.MODE_NOWRAP);

        //Display the user's sessionID
        p.addChild(new Text("Session: "+ sessionID));
        p.addChild(new Break());

        //Display the user's hit count
        p.addChild(new Text("Hit #:" + hitNumber.toString()));
        p.addChild(new Break());

        //Create a random parameter to be added to the URL later
        //Adding the empty string to the end of Math.random() converts the result from a    ↙
            double to a string
        String rnd = Math.random() + "";
        rnd = rnd.substring(2,6);

        //Build the URL...
        //Some devices cache content more than they should. Adding a random parameter is an  ↙
            unbeautiful,
        //though often necessary technique for tricking the phone into always hitting the    ↙
            server rather
        //than getting a page from its local cache.
        //n.b. Certain phones (such as the Nokia WAP Toolkit Version 2.0 simulator) require  ↙
            the absolute
        //application path so we include it here
        String href = appPath+ "?sid=" + sessionID + "&amp;rnd=" +rnd;

        //Constructs a Go tag with the appropriate URL and link method.
        Go go = new Go(href,true,Go.METHOD_GET);

        //Specifies the action to perform when the user activates the link and the text the  ↙
            device will
        //Display to represent the link.
        Anchor anchor = new Anchor(go,new Text("Hit again..."));

        //Add the anchor to the Paragraph
        p.addChild(anchor);

        //Add the Paragraph to the card
        card.addParagraph(p);

        //Add the card to the deck
        deck.addCard(card);

        //Render it (that is, turn it into a String)
        String resultString = deck.render();

        //Returns the entire rendered document text, suitable for display in a WML browser
        return resultString;
    }
}
```

```
===============================================================================

                    Logging Connector Sample Connector

                    Wireless SDK for ThinAir Server

===============================================================================


-----------------
About this Sample
-----------------

This connector demonstrates the logging capabilites of the ThinAir Server.  It
is a HTML only Connector


------------
Requirements
------------

This sample requires the following SDK JARs:

    * platform.jar

    * taglib.jar


------------
Sample Files
------------

This sample consists of the following file tree:

    connector.ini - connector configuration file

    LoggingConnector.class - compiled Java code

    /src - java source file - DBconnector.java


-------------------
Building the Sample
-------------------

Compile the sample code using the Java compiler of your choice.  Make sure to
append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a
subdirectory of the ThinAir Server's /Connectors subdirectory, given a name
of your choice.

The logging API's uses settings in connector.ini file to place the log entries.
Your connector.ini must have an [Output] section followed by
[level] = [destination].

Example:
[Output]
Critical = STDERR, FILE:logs\CriticalLog.txt
Error = STDERR, FILE:logs\ErrorLog.txt
Warning = STDOUT, FILE:logs\WarningLog.txt
Info = STDOUT, FILE:logs\InfoLog.txt

Consult the Developers guide for more information

Start the ThinAir Server, it will load the sample code and begin executing it.


----------------
Using the Sample
----------------

Wait until the ThinAirServer has started and the DBconnector has
been loaded and initialized.  From your wireless WML device, or web browser,
enter the IP address listed as the value for ApplicationPath in connector.ini
(your ThinAirServer IP address), followed by /samples/LoggingConnector.  For
a machine with IP address 111.222.12.34 this would be:
     http://111.222.12.34/samples/LoggingConnector

Follow the on-screen instructions.

==============================================================================
                        Last updated: 11.13.2000
                  Copyright 1999, 2000 ThinAirApps Inc.
==============================================================================
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//Standard ThinAir server imports
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.tag.html.*;

//Standard Java Imports
import java.util.*;
import java.io.*;

/**
 * This Connector is for demonstrating the Logging capabilities of the ThinAir Server.  The ↙
    user can select
 * which log file to write to.  The location of the log file is determined by the connector. ↙
    ini
 * settings.  Please refer to the JavaDocs for a comprehensive list of the methods available.
 *
 */
public class LoggingConnector implements Connector
{
    ApplicationLog appLog;
    String appPath;



   /**init() is called by the ThinAirServer when the Connector is loaded.  It provides the    ↙
    Connector with
     * resources it needs to interact with the ThinAirServer.
     * For more information about the Connector interface, see the javadocs for the ThinAir  ↙
        Server API
     *
     * @param appName is a String derived from connector.ini.  We used this to format our    ↙
        action field in the form tag
     * @param ap is a String derived from connector.ini.  We don't need this for this sample.
     * @param props is a Properties list containing developer assigned connector-specific    ↙
        properties.
     * We don't need this parameter in this sample.
     * @param connectorAccess is our access point to the services provided by ThinAir Server ↙
        .  We don't need this for this sample.
     * @param al is the Application Log instance that we use to write to the appropiate logs ↙
        .  We used this to write to logs
     */
    public void init(String appName, String ap, Properties props, ConnectorAccess ca,        ↙
        ApplicationLog al)
    {
        //Set the two values
        appLog = al;
        appPath = ap;
    }



    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a        ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing  ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used  ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this     ↙
        Connector supports.
     */
    public String[] getDevices()
    {

        String devices[] = { "TA_HTML" };
        return devices;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming     ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,     ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param reqprops a set of name value pairs corresponding to the HTTP request parameters↙
        from the device.
     * @param dev a Device object created in the image of the actual device making this      ↙
        request.
     * @param out a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties reqProps, Device dev, OutputStream out)
    {
        String result;
        int actionCode;
        String sactcode;

        //Get the actionCode from the form, if this the first time through, then getProperty ↙
            will return null
        sactcode = reqProps.getProperty("actionCode");

        //Since this is the first time through we do not need to check, so we set actionCode ↙
            to zero
        if (sactcode == null)
            actionCode = 0;
        else
            //otherwise we parse actionCode into a int
            actionCode = Integer.parseInt(sactcode);
        Form mainForm;
        String msgString = "";

        //Start Generating HTML
        HTMLTagDocument htmlDoc = new HTMLTagDocument();
        Head htmlHead = new Head();
        htmlDoc.addChild(htmlHead);
        Body htmlBody = new Body();

        //check whether actionCode has been set
        if (actionCode != 0)
        {
            //test actionCode
            switch (actionCode)
            {
```

```
            //Using the instance of ApplicationLog that was passed in the init method
            //and depending on what the actionCode is, we call the appropiate logger ↙
                method.
            //The logging methods uses the following conventions:
            //logXXX(<string of method call>, <error code>, <message>, <boolean of ↙
                whether or not to display the connector name>)
            //where XXX is the log Level name (Emergency, Critical, Alert, Error, Warning↙
                , Notice, Info, Debug)
            //for more detail information, please refer to the Developers Guide
        case 1:
            appLog.logEmergency("handle()", 1100, "Emergency message-Indicate fatal ↙
                conidition, probably resulting in exception every time code is excuted", ↙
                true);
            msgString = "An Emergency message was written to the location specified in ↙
                connector.ini";
            break;
        case 2:
            appLog.logCritical("handle()", 2100, "Critical message-Critical conditions, ↙
                such as hard device errors", false);
            msgString = "A Critical message was written to the location specified in ↙
                connector.ini";
            break;
        case 3:
            appLog.logAlert("handle()", 3100, "Alert message-A condition to be corrected ↙
                immediately, such as a corrupt system", true);
            msgString = "An Alert message was written to the location specified in ↙
                connector.ini";
            break;
        case 4:
            appLog.logError("handle()", 4100, new Exception("Throwing Exception for ↙
                Logging Error"), true);
            msgString = "An Exception was thrown to the location specified in connector.↙
                ini";
            break;
        case 5:
            appLog.logWarning("handle()", 5100, "Warning message-Indicate an unusual ↙
                condition that the app will handle automatically but which is noted in ↙
                the log to help diagnose futre errors", false);
            msgString = "A Warning message was written to the location specified in ↙
                connector.ini";
            break;
        case 6:
            appLog.logNotice("handle()", 6100, "Notice message-Conditions that are not ↙
                errors, but may require special handling", false);
            msgString = "A Notice message was written to the location specified in ↙
                connector.ini";
            break;
        case 7:
            appLog.logInfo("handle()", 7100, "Info message-Normal Status message", true);
            msgString = "An Info message was written to the location specified in ↙
                connector.ini";
            break;
        case 8:
            appLog.logDebug("handle()", 8100, "Debug message-for logging debug msgs ↙
                during development process", false);
            msgString = "A Debug message was written to the location specified in ↙
                connector.ini";
            break;
        default:
            break;
        }
        //Add the msg to HTML
        htmlBody.addChild(new Text(msgString));
        htmlBody.addChild(new Break());
    }

    //Construct the mainSelection Form
    mainForm = renderMainSelection();
    htmlBody.addChild(mainForm);
```

```
        htmlDoc.addChild(htmlBody);

        //Render the HTML
        result = htmlDoc.render();
        try
        {
            //Write out
            out.write(result.getBytes());
        }
        catch (Exception e)
        {
            //Catch the exception
            appLog.logError("handle()", 4100, "Error writing to Outputstream: " + e.
                getMessage(), true);
        }

    }

    //This function creates the Form using the HTML tag libraries
    public Form renderMainSelection()
    {

        Form formtag = new Form("Logging" , appPath, "GET" );
        formtag.addChild(new Text("Please select which Log you would like to write to:"));
        formtag.addChild(new Break());
        formtag.addChild(new Text("Emergency"));
        Input emergencyInput = new Input("radio", "actionCode", "1");
        formtag.addChild(emergencyInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Critical"));
        Input alertInput = new Input("radio", "actionCode", "2");
        formtag.addChild(alertInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Alert"));
        Input criticalInput = new Input("radio", "actionCode", "3");
        formtag.addChild(criticalInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Error"));
        Input errorInput = new Input("radio", "actionCode", "4");
        formtag.addChild(errorInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Warning"));
        Input warnInput = new Input("radio", "actionCode", "5");
        formtag.addChild(warnInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Notice"));
        Input noticeInput = new Input("radio", "actionCode", "6");
        formtag.addChild(noticeInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Info"));
        Input infoInput = new Input("radio", "actionCode", "7");
        formtag.addChild(infoInput);
        formtag.addChild(new Break());

        formtag.addChild(new Text("Debug"));
        Input debugInput = new Input("radio", "actionCode", "8");
        formtag.addChild(debugInput);
        formtag.addChild(new Break());

        formtag.addChild(new SubmitButton("Submit"));

        return formtag;
        }
}
```

09574328.050501

HTML Rendering Sample Connector

Wireless SDK for ThinAir Server

About this Sample

This sample Connector demonstrates the use of the HTML Tag library to accept input from HTML forms and render output for HTML browsers.

Requirements

This sample requires the following SDK JARs:

  * platform.jar

  * taglib.jar

This sample does not require any other external APIs.

Sample Files

This sample consists of the following file tree:

```
   connector.ini - connector configuration file

   HTMLRendererConnector.class - compiled Java code

   /src - java source file - HTMLRendererConnector.java
```

Building the Sample

Compile the sample code using the Java compiler of your choice. Make sure to append the required jar files above into your CLASSPATH.

Install the compiled sample code and connector.ini configuration file into a subdirectory of the ThinAir Server's /Connectors subdirectory, given a name of your choice.

Start the ThinAir Server, it will load the sample code and begin executing it.

Using the Sample

Wait until the ThinAirServer has started and the HTML Rendering Connector has been loaded and initialized. From your wireless HTML device, or web browser, enter the IP address listed as the value for ApplicationPath in connector.ini (your ThinAirServer IP address), followed by /samples/html. For a machine with IP address 111.222.12.34 this would be:

    http://111.222.12.34/samples/html

Follow the on-screen instructions.

Last updated: 11.13.2000

```
/**
 * @(#)HTMLRendererConnector
 *
 * Copyright (c) 2000 ThinAirApps, Inc. All Rights Reserved
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 */



//ThinAir Platform import
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.device.*;

//ThinAir Tag Libraries import
import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;

//Standara Java import
import java.util.*;
import java.io.*;




/**
 * This is a simple sample whose purpose is to illustrate the use of the ThinAir HTML
 * Tag Library.  It creates two simple HTML decks: the first prompts the user for their
 * favorite color, username and password, the second simply echos the submitted values.
 * This sample makes use of HTMLTagDocument, Body, Form, LabeledInput, PasswordField, Select,
 * Option, SubmitButton, Table, TableRow and TableCell.
 *
 * For a comprehensive reference of the HTML Tag Library, see the ThinAir Javadoc API ↙
   documentation.
 */
public class HTMLRendererConnector implements Connector
{

    String          path;
    //The friendly name of this sample app
    String          appName;
    //Our access point to the services of ThinAir Server
    ConnectorAccess access;



    /**
     * init() is called by the ThinAirServer when the Connector is loaded.  It provides the
     * Connector with resources it needs to interact with the ThinAirServer.
     *
     * For more information about the Connector interface, see the ThinAir Javadoc API ↙
       documentation.
     *
     * @param applicationName is a String derived from connector.ini.  We don't need this for↙
       this sample.
     * @param applicationPath is a String dervid from connector.ini.  We don't need this for ↙
       this sample.
     * @param connectorProps is a Properties list containing developer assigned ↙
       connector-specific properties.
     * We don't need this parameter in this sample.
     * @param connectorAccess is our access point to the services provided by ThinAir Server ↙
       .  We don't need this for this sample.
     * @param ApplicationLog is used for Logging.  We do not use this parameter in this ↙
       sample
     */
    public void init(String applicationName, String applicationPath, Properties ↙
        connectorProps, ConnectorAccess connectorAccess, ApplicationLog al)
    {
        appName = applicationName;
        access = connectorAccess;
        path = applicationPath;
    }



    /**
     * getDevices() is called once by the ThinAir Server during start-up.  It allows a ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String deviceType = "TA_HTML";
        String deviceTypes[] = { deviceType };
        return deviceTypes;
    }



    /**
     * The handle method implements the core logic of a Connector.  It takes an incoming ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports, ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters ↙
        from the device.
     * @param device a Device object created in the image of the actual device making this ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream result) throws ↙
        IOException
    {

        String resultString = null;

        //get the 'action' parameter from the request.
        //This is an HTTP param we define to determine what action to take when we get a ↙
```

```
            request.
        String action = props.getProperty("a");

        //if this is the first hit (or any request for the main deck)
        if (action == null)
        {
            // build a deck that lets the user enter information.
            resultString = renderWelcome();

        }
        //if they have already entered the information, then display it...
        else if (action.equals("display"))
        {
            //build a display deck with the entered info, pass the request properties in...
            resultString = renderAnswers(props);
        }

        result.write(resultString.getBytes());
    }



    /**
     * This method renders a page with a form for entering information...
     *
     * @return the rendered page.
     */
    private String renderWelcome()
    {
        //create the page
        HTMLTagDocument doc = new HTMLTagDocument();

        //create the body
        Body body = new Body();

        //set the background color
        body.addAttribute("bgcolor","#ffffff");

        Bold bold = new Bold();

        //add a title
        bold.addChild(new Text("Render HTML Connector: Welcome"));

        body.addChild(bold);

        //create a input Form
        Form form = new Form ("render", path + "?a=display", "POST");

        form.addChild(new HorizontalRule());

        //Create a dropdown list...
        Select colorSelect = new Select ("color");

        //create some choices
        Option red = new Option ("r","Red");
        Option orange = new Option ("o","Orange");
        Option yellow = new Option ("y","Yellow");
        Option green = new Option ("g","Green");
        Option blue = new Option ("b","Blue");
        Option indigo = new Option ("i","Indigo");
        Option plaid = new Option ("p","Plaid");

        //add the choices
        colorSelect.addOption(red);
        colorSelect.addOption(orange);
        colorSelect.addOption(yellow);
        colorSelect.addOption(green);
        colorSelect.addOption(blue);
        colorSelect.addOption(indigo);
        colorSelect.addOption(plaid);

        Bold colorLbl = new Bold();
        colorLbl.addChild(new Text("Favorite color: "));

        //add a label
        form.addChild(colorLbl);

        //add the select to the Form
        form.addFormElement(colorSelect);
        form.addChild(new Break());

        //add an input field
        form.addFormElement(new LabeledInput("usr", "Username: "));

        form.addChild(new Break());

        //create a password field
        PasswordField pwdInput = new PasswordField("pwd");

        //add a label
        form.addChild(new Text("Password: "));

        //add the password field
        form.addFormElement(pwdInput);
        form.addChild(new Break());

        form.addChild(new HorizontalRule());
        form.addChild(new Break());

        //add a button...
        form.addFormElement(new SubmitButton ("Next"));

        body.addChild(form);

        doc.addChild(body);

        String resultString = doc.render();

        return resultString;
    }



    /**
     * Create a page with the results of the query above
     *
     * @param props Properties of user responses
     *
     * @return String of HTML page for display
     */
    private String renderAnswers(Properties props)
    {
        //get the arguments passed from the welcome deck
        String usrName = props.getProperty("usr");
        String password = props.getProperty("pwd");

        //this will be a single letter, so we have to map it to a color
        String color = props.getProperty("color");
        if (color == null)
            return renderException("Error! No color was entered.");
        if (color.equals("r"))
            color = "Red";
        else if (color.equals("o"))
            color = "Orange";
        else if (color.equals("y"))
            color = "Yellow";
        else if (color.equals("g"))
            color = "Green";
        else if (color.equals("b"))
            color = "Blue";
        else if (color.equals("i"))
            color = "Indigo";
        else if (color.equals("p"))
            color = "Plaid";

        //create the page
        HTMLTagDocument page = new HTMLTagDocument();

        Body body = new Body();

        //set the background color
        body.addAttribute("bgcolor","#ffffff");

        //create a new Table with a thin border
        Table table = new Table(1);

        //create a TableRow
        TableRow tr = new TableRow();

        //create a TableCell
        TableCell tc = new TableCell();

        //add the text
        tc.addChild(new Text("Color: "));

        //add the cell to the row
        tr.addChild(tc);

        //make a new cell
        tc = new TableCell();

        //add the text
        tc.addChild(new Text(color));

        //add the cell to the row
        tr.addChild(tc);

        //add the row to the table
        table.addChild(tr);

        //continue this for each row...
        tr = new TableRow();
        tc = new TableCell();
        tc.addChild(new Text("Username: "));
        tr.addChild(tc);

        tc = new TableCell();
        tc.addChild(new Text(usrName));
        tr.addChild(tc);
        table.addChild(tr);

        tr = new TableRow();
        tc = new TableCell();

        //Show the password for this sample. Of course, this is not recommended...
        tc.addChild(new Text("Password: "));
        tr.addChild(tc);

        tc = new TableCell();
        tc.addChild(new Text(password));
        tr.addChild(tc);
        table.addChild(tr);

        //add the table to the page's body
        body.addChild(table);

        //add the body to the page
        page.addChild(body);

        //render the page
        String resultString = page.render();

        return resultString;

    }


    /**
     * This is a simple exception rendering method.
     *
     * @param message the message to be presented to the user
     *
     * @return the rendered HTML page deck
     */
    private String renderException (String message)
    {
        //create the page
        HTMLTagDocument page = new HTMLTagDocument();

        Body body = new Body();

        //set the background color
        body.addAttribute("bgcolor","#ffffff");

        body.addChild(new Text(message));
        body.addChild(new Break());

        String resultString = body.render();

        return resultString;
    }
}
```

```
import com.thinairapps.platform.connector.*;
import com.thinairapps.tag.html.*;
import com.thinairapps.platform.device.*;

import java.util.*;
import java.io.*;

/**
 *
 * Copyright (c) 2000 ThinAirApps, Inc. All Rights Reserved.
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 * The basis of this sample application is to demonstrate the "portal" example  where there ↙
   is a
 * single log on and then the user can proceed to other "applications" within the portal.  In↙
   this
 * case the user logs in and then is presented with a menu of 2 other applications.  When the↙
   user
 * enters one of the applications it then checks to see if it has been configured yet, if it ↙
   has not
 * it will then present the configuration fields, the user enters and proceeds, the         ↙
   application then displays
 * the information entered.  At this time the user can either go back to the original       ↙
   application, where it will display
 * the data entered previously or proceed to the second app and the same process repeats.   ↙
   This demonstrates how a developer
 * can develop numerous apps with a single log in.  The main detail being the ability to pass↙
   the Session ID around and
 * use it for each application to store data (in this case the user ID and password)
 *  This connector is the main connector that leads to the other "apps".  It uses Sessions  ↙
   and User Profiles.  Sessions
 * are temporary while User Profile is stored and can be retrieved at a later time.
 */

public class PortalConnector implements Connector
{
    ConnectorAccess myCA;
    String appPath;

    //need to store the ConnectorAccess and ApplicationPath
    public void init(String name, String p, Properties iniProps, ConnectorAccess ca,        ↙
        ApplicationLog al)
    {
        myCA = ca;
        appPath = p;
    }

    //this connector only supports HTML
    public String[] getDevices()
    {
        String[] devices = {"TA_HTML"};
        return devices;
    }

    public void handle(Properties reqProps, Device device, OutputStream out) throws         ↙
        IOException
    {
        String sid = null;
        String result = null;
        String firstTime;
        Hashtable cache = null;

        String login;
        String passwd;
        //these variables are used to keep track of when a user enters and what screen should↙
            be presented
        firstTime = reqProps.getProperty("firstTime");
        sid = reqProps.getProperty("sid");
        if (firstTime == null)
        {
            //If it is the firstTime then
            //create a new session
            result = generateLogin();
        }
        else
        {
            //It's not the first time,but I still haven't generated a Session ID yet
            if (sid == null)
            {
                try
                {
                //Create the session using ConnectorAccess
                sid = myCA.createSession();
                //get the cache for this session
                cache = myCA.getSessionCache(sid);
                }
                catch (Exception e)
                {
                //Can pass any exception messages to the exception rendering method
                }

                //As long as I get the cache, then proceed to populate with data
                if (cache != null)
                {
                login = reqProps.getProperty("usr");
                passwd = reqProps.getProperty("pwd");
                cache.put("usr",login);
                cache.put("pwd", passwd);
                }
            }
            //The values are stored in the cache, display the menu to the user
            result = generateMenu(sid);
        }


            out.write(result.getBytes());

    }

    public String generateLogin()
    {
        //Generate the HTML for the User to Login
        HTMLTagDocument htmlTag = new HTMLTagDocument();
        Body bodyTag = new Body();

        Paragraph pTag = new Paragraph();

        pTag.addChild(new Text("Welcome to the Portal"));
        pTag.addChild(new Break());
        pTag.addChild(new Text("Please Login"));
        pTag.addChild(new Break());
        bodyTag.addChild(pTag);

        Form formTag = new Form ("Login", appPath, "POST");

        formTag.addFormElement(new LabeledInput("usr", "Username: "));

        formTag.addChild(new Break());

        PasswordField pwdInput = new PasswordField("pwd");  //create a password field

        formTag.addChild(new Text("Password: "));  //add a label
        formTag.addFormElement(pwdInput);          //ad the password field
        formTag.addChild(new Break());
        formTag.addChild(new HiddenInput("firstTime","Yes"));
        formTag.addChild(new SubmitButton ("Submit"));

        bodyTag.addChild(formTag);

        htmlTag.addChild(bodyTag);

        return htmlTag.render();
    }

    public String generateMenu(String sid)
    {
        //generate the Menu
        HTMLTagDocument htmlTag = new HTMLTagDocument();
        Body bodyTag = new Body();

        Paragraph pTag = new Paragraph();

        pTag.addChild(new Text("Please select the application you woud like"));
        pTag.addChild(new Break());

        bodyTag.addChild(pTag);

        Anchor an1 = new Anchor("Application1", "/portal/app1?sid=" + sid, new Text ↙
            ("Application 1"));
        Anchor an2 = new Anchor("Application2", "/portal/app2?sid=" + sid, new Text ↙
            ("Application 2"));
        bodyTag.addChild(an1);
        bodyTag.addChild(new Break());
        bodyTag.addChild(an2);
        bodyTag.addChild(new Break());


        htmlTag.addChild(bodyTag);

        return htmlTag.render();
    }
}
```

```
/**
 *
 * Copyright (c) 2000 ThinAirApps, Inc. All Rights Reserved.
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 * Here is just a class that has some properties and methods, the main thing to mention is ↙
   that
 * it implements Serializable, this is required to store data in UserProfile
 */

public class Application1UserData implements java.io.Serializable
{
    private String ssNum;
    private String password;
    private String host;
    private String additionalParam;

    public Application1UserData()
    {
        ssNum="Unknown";
        password="None";
        host="n/a";
        additionalParam="n/a";
    }

    public Application1UserData(String ssn, String pw, String ht, String ap)
    {
        ssNum = ssn;
        password = pw;
        host = ht;
        additionalParam = ap;
    }

    public void setssNum(String ssn)
    {
        ssNum = ssn;
    }

    public void setPassword(String pw)
    {
        password = pw;
    }

    public void setHost(String ht)
    {
        host = ht;
    }

    public void setAdditionalParam(String ap)
    {
        additionalParam = ap;
    }

    public String getssNum()
    {
        return ssNum;
    }

    public String getPassword()
    {
        return password;
    }

    public String getHost()
    {
        return host;
    }

    public String getAddditionalParam()
    {
        return additionalParam;
    }
}
```

```
import com.thinairapps.platform.connector.*;
import com.thinairapps.tag.html.*;
import com.thinairapps.platform.device.*;

import java.util.*;
import java.io.*;


/**
 *
 * Copyright (c) 2000 ThinAirApps, Inc. All Rights Reserved.
 *
 * ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE LICENSE     ↙
   AGREEMENT
 * BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR USE OF THE SOFTWARE IN VIOLATION OF ↙
   THE
 * SOFTWARE LICENSE AGREEMENT IS STRICTLY PROHIBITED.
 *
 * This Connector is called from the main menu of the portal.  It first checks to see if the ↙
   session passed in is valid
 * Then it checks to see if the user is in User Profile, if their is an entry then display   ↙
   the data, if there isn't
 * then display the configuration field and have the user enter the data.  There is a bit of ↙
   a tricky part and the way
 * I've implemented is not the most robust way.  The tricky part deals with if a user already↙
   has a User Profile, but from
 * another application.  In that case it needs to check if the data in User profile is     ↙
   associated with this application
 * in particular.  I've marked the code where the logic is taking place
 */
public class Application1Connector implements Connector
{
    ConnectorAccess app1CA;
    String app1Path;
    String app1Name;

    public void init(String appName, String appPath, Properties connectorProps,         ↙
        ConnectorAccess ca, ApplicationLog al)
    {
        app1CA = ca;
        app1Path = appPath;
        app1Name = appName;
    }

    public String[] getDevices()
    {
        return new String[] {"TA_HTML"};
    }

    public void handle(Properties appProps, Device dev, OutputStream out)
    {
        String sid;
        String usrName;
        String password;
        String result;
        String dataEntered;

        boolean valid;
        boolean userExists;
        Hashtable cache = null;

        //first get the SID, in this case the only way that a user should enter this       ↙
            application is through the menu

        sid = appProps.getProperty("sid");

        //another variable to see where the user is

        dataEntered = appProps.getProperty("dataEntered");
        //check to see if the session is valid, it might have timed out....
        valid = app1CA.sessionValid(sid);

        if (valid)
        {
            //extract user information
            //Now get the cache from the SID
            try
            {
            cache = app1CA.getSessionCache(sid);
            }
            catch (Exception e)
            {
                //catch NoSuchSessionException
            }

            //retrieve the username and password
            usrName = (String)cache.get("usr");
            password = (String)cache.get("pwd");
            //Here's the TRICKY part, now we go and look to see if there is data associated ↙
                with this
            //application in particular.  The method userProfileExists does not tell us which↙
                application
            //(if any) there is data for.
            userExists = app1CA.userProfileExists(usrName);

            Application1UserData app1UserEnteredData = null;
            try
            {
                //Here is where we try to see if there is data associated with this         ↙
                    application
                app1UserEnteredData = (Application1UserData)app1CA.getUserProfileData(usrName↙
                    , app1Name);
            }
            catch(Exception e)
            {
                //catch NoSuchUserProfileException
            }
            //Here we check 2 things
            //1)If the user does not exist, then it is his first time and display the       ↙
                configuration screen
            //2) Or he already exists, but the data is for another application, display the ↙
                configuration screen for htis application
            if (!userExists || app1UserEnteredData == null)
            {
                //check to see if data for user has been entered
                //this needs to be passed twice, once to check if user exists, then check    ↙
                    again for entering data
                if (dataEntered != null)
                {
                //store user data
                //instatiate Applicaiton1UserData
                Application1UserData app1UserData = new Application1UserData();
                //Set the variables
                app1UserData.setssNum(appProps.getProperty("ssNum"));
                app1UserData.setPassword(appProps.getProperty("pwd"));
                app1UserData.setHost(appProps.getProperty("host"));
                app1UserData.setAdditionalParam(appProps.getProperty("additionalParam"));

                //1st create the user profile
                try
                {
                    //Here's another check, if the user already exists, and we are here that ↙
                        means
                    //there was no data for this application, but we do not need to create   ↙
                        another userprofile
                    //so if the User does not exist, then create....
                    if (!userExists)
                {
                    app1CA.createUserProfile(usrName, password, app1Name);
                }
                //2nd now add the application data
                app1CA.setUserProfileData(usrName, app1Name, app1UserData);
            }
            catch (Exception e)
            {
                //catch exception for ProfileAlready Exists
                //catch exception for ProfileStoreLockedException
            }

            //Some HTML to tell the user that the data has been stored
            HTMLTagDocument htmlTag = new HTMLTagDocument();
            Body bodyTag = new Body();

            Paragraph pTag = new Paragraph();

            pTag.addChild(new Text("Application 1 User Added"));
            pTag.addChild(new Break());
            pTag.addChild(new Text("For User " + usrName));
            pTag.addChild(new Break());
            bodyTag.addChild(pTag);

            //View the data entered, incorporate the SID into the URL
            Anchor an1 = new Anchor("View Data", app1Path + "?sid=" + sid, new Text("View↙
                Data"));
            bodyTag.addChild(an1);
            htmlTag.addChild(bodyTag);
            try
            {
            out.write( htmlTag.render().getBytes());
            }
            catch (Exception e)
            {
                //catch out exception
            }


            }
            else
            {
                //Else the user does not exist so display the configuration screen
                result = renderInputUserDataScreen(sid);
                try
                {
                out.write(result.getBytes());
                }
                catch (Exception e)
                {
                    //catch IO excecption
                }
            }
        }
        else
        //if user exists and Application Data for this app exists that means that data    ↙
            already is there
        //so display data to user
        {
                HTMLTagDocument htmlTag = new HTMLTagDocument();
                Body bodyTag = new Body();

                Paragraph pTag = new Paragraph();

                pTag.addChild(new Text("Application 1 Data"));
                pTag.addChild(new Break());
                pTag.addChild(new Text("For User " + usrName));
                pTag.addChild(new Break());
                bodyTag.addChild(pTag);
```

```
                    bodyTag.addChild(new Text("Social Number: " + appl1UserEnteredData.     ↙
                        getssNum()));
                    bodyTag.addChild(new Break());
                    bodyTag.addChild(new Text("Password: " + appl1UserEnteredData.getPassword ↙
                        ()));
                    bodyTag.addChild(new Break());
                    bodyTag.addChild(new Text("Host: " + appl1UserEnteredData.getHost()));
                    bodyTag.addChild(new Break());
                    bodyTag.addChild(new Text("Additional Parameter: " + appl1UserEnteredData.↙
                        getAddditionalParam()));
                    bodyTag.addChild(new Break());
                    bodyTag.addChild(new Text("This data is now stored in UserProfile and can↙
                        be retrieved at any time with the Login and Password<br>"));
                    Anchor an1 = new Anchor("Menu", "/portal?firstTime=1&sid=" + sid, new     ↙
                        Text("Return to Portal Menu"));
                    bodyTag.addChild(an1);
                    htmlTag.addChild(bodyTag);
                    try
                    {
                    out.write( htmlTag.render().getBytes());
                    }
                    catch (Exception e)
                    {
                        //catch out exception
                    }

            }

        }
        else
        {
            String result2 = "Sorry Session is not valid or Timed Out, please login again";
            try
            {
            out.write(result2.getBytes());
            }
            catch (Exception e)
            {
            }

        }

    }

    public String renderInputUserDataScreen(String sid)
    {
        HTMLTagDocument htmlTag = new HTMLTagDocument();
        Body bodyTag = new Body();

        Paragraph pTag = new Paragraph();

        pTag.addChild(new Text("Please configure Application 1"));
        pTag.addChild(new Break());
        pTag.addChild(new Text("Enter Data"));
        pTag.addChild(new Break());
        bodyTag.addChild(pTag);

        Form formTag = new Form ("UserData", appl1Path, "POST");

        formTag.addFormElement(new LabeledInput("ssNum", "Social Security Number: "));

        formTag.addChild(new Break());

        PasswordField pwdInput = new PasswordField("pwd");

        formTag.addChild(new Text("Password: "));
        formTag.addFormElement(pwdInput);
        formTag.addChild(new Break());
        formTag.addFormElement(new LabeledInput("host", "Host: "));
        formTag.addChild(new Break());
        formTag.addFormElement(new LabeledInput("additionalParam", "Additional Parameter   ↙
            : "));
        formTag.addChild(new Break());
        formTag.addChild(new HiddenInput("sid",sid));
        formTag.addChild(new HiddenInput("dataEntered","Yes"));
        formTag.addChild(new SubmitButton ("Submit"));

        bodyTag.addChild(formTag);

        htmlTag.addChild(bodyTag);

        return htmlTag.render();

    }
}
```

```
        catch(Exception e)
        {
            //catch NoSuchUserProfileException
        }
        if (!userExists || app2UserEnteredData == null)
        {

                //check to see if data for user has been entered
                //this needs to be passed twice, once to check if user exists, then check↙
                    again for entering data
                if (dataEntered != null)
                {
                    //store user data
                    //instatiate Applicaiton1UserData
                    Application2UserData app2UserData = new Application2UserData();
                    //Set the variables
                    app2UserData.setPreferences(appProps.getProperty("preferences"));
                    app2UserData.setAge(Integer.parseInt(appProps.getProperty("age")));


                    //1st create the user profile
                    try
                    {
                        if(!userExists)
                        {
                            app2CA.createUserProfile(usrName, password, app2Name);
                        }
                        //2nd now add the application data
                        app2CA.setUserProfileData(usrName, app2Name, app2UserData);
                    }
                    catch (Exception e)
                    {
                        //catch exception for ProfileAlready Exists
                        //catch exception for ProfileStoreLockedException
                    }


                    HTMLTagDocument htmlTag = new HTMLTagDocument();
                    Body bodyTag = new Body();

                    Paragraph pTag = new Paragraph();

                    pTag.addChild(new Text("Application 2 User Added"));
                    pTag.addChild(new Break());
                    pTag.addChild(new Text("For User " + usrName));
                    pTag.addChild(new Break());
                    bodyTag.addChild(pTag);
                    Anchor an1 = new Anchor("View Data", app2Path + "?sid=" + sid, new   ↙
                        Text("View Data"));
                    bodyTag.addChild(an1);
                    htmlTag.addChild(bodyTag);
                    try
                    {
                        out.write( htmlTag.render().getBytes());
                    }
                    catch (Exception e)
                    {
                        //catch out exception
                    }
                }
                else
                {
                    result = renderInputUserDataScreen(sid);
                    try
                    {
                    out.write(result.getBytes());
                    }
                    catch (Exception e)
                    {
                        //catch IO excecption
                    }
                }
        }
        else
        //if user exists that means that data already is there
        //so display data to user
        {
                HTMLTagDocument htmlTag = new HTMLTagDocument();
                Body bodyTag = new Body();

                Paragraph pTag = new Paragraph();

                pTag.addChild(new Text("Application 2 Data"));
                pTag.addChild(new Break());
                pTag.addChild(new Text("For User " + usrName));
                pTag.addChild(new Break());
                bodyTag.addChild(pTag);

                bodyTag.addChild(new Text("Preferences: " + app2UserEnteredData.   ↙
                    getPreferences()));
                bodyTag.addChild(new Break());
                bodyTag.addChild(new Text("Age: " + app2UserEnteredData.getAge()));
                bodyTag.addChild(new Break());
                bodyTag.addChild(new Text("This data is now stored in UserProfile and can↙
                    be retrieved at any time with the Login and Password<br>"));
                Anchor an1 = new Anchor("Menu", "/portal?firstTime=1&sid=" + sid, new    ↙
                    Text("Return to Portal Menu"));
                bodyTag.addChild(an1);
                htmlTag.addChild(bodyTag);
                try
                {
                out.write( htmlTag.render().getBytes());
                }
                catch (Exception e)
                {
                    //catch out exception
                }
        }

    }
    else
    {
        String result2 = "Sorry Session is not valid or Timed Out, please login again";
        try
        {
        out.write(result2.getBytes());
        }
        catch (Exception e)
        {
        }

    }


}

public String renderInputUserDataScreen(String sid)
{
    HTMLTagDocument htmlTag = new HTMLTagDocument();
    Body bodyTag = new Body();

    Paragraph pTag = new Paragraph();

    pTag.addChild(new Text("Please configure Application 1"));
    pTag.addChild(new Break());
    pTag.addChild(new Text("Enter Data"));
    pTag.addChild(new Break());
    bodyTag.addChild(pTag);
```

```
        Form formTag = new Form ("UserData", app2Path, "POST");

        formTag.addFormElement(new LabeledInput("preferences", "Preferences: "));

        formTag.addChild(new Break());

        formTag.addFormElement(new LabeledInput("age", "Age: "));
        formTag.addChild(new Break());
        formTag.addChild(new HiddenInput("sid",sid));
        formTag.addChild(new HiddenInput("dataEntered","Yes"));
        formTag.addChild(new SubmitButton ("Submit"));

        bodyTag.addChild(formTag);

        htmlTag.addChild(bodyTag);

        return htmlTag.render();

    }
}
```

```
/**
 * Here is just a class that has some properties and methods, the main thing to mention is ↙
    that
 * it implements Serializable, this is required to store data in UserProfile
 */

public class Application2UserData implements java.io.Serializable
{
    private String preferences;
    private int age;

    public Application2UserData()
    {
        preferences = "None";
        age = 0;
    }

    public Application2UserData(String p, int a)
    {
        preferences = p;
        age = a;
    }

    public void setPreferences(String p)
    {
        preferences = p;
    }

    public void setAge(int a)
    {
        age = a;
    }

    public String getPreferences()
    {
        return preferences;
    }

    public int getAge()
    {
        return age;
    }
}
```

```
import com.thinairapps.platform.connector.*;
import com.thinairapps.tag.html.*;
import com.thinairapps.platform.device.*;

import java.util.*;
import java.io.*;
/**
 * This is the same as for Application 1, except different set of data for Application 2
 */
public class Application2Connector implements Connector
{
    ConnectorAccess app2CA;
    String app2Path;
    String app2Name;

    public void init(String appName, String appPath, Properties connectorProps,  ↙
        ConnectorAccess ca, ApplicationLog al)
    {
        app2CA = ca;
        app2Path = appPath;
        app2Name = appName;
    }

    public String[] getDevices()
    {
        return new String[] {"TA_HTML"};
    }

    public void handle(Properties appProps, Device dev, OutputStream out)
    {
        String sid;
        String usrName;
        String password;
        String result;
        String dataEntered;

        boolean valid;
        boolean userExists;
        Hashtable cache = null;

        sid = appProps.getProperty("sid");

        dataEntered = appProps.getProperty("dataEntered");

        valid = app2CA.sessionValid(sid);

        if (valid)
        {
            //extract user information
            try
            {
            cache = app2CA.getSessionCache(sid);
            }
            catch (Exception e)
            {
                //catch NoSuchSessionException
            }

            usrName = (String)cache.get("usr");
            password = (String)cache.get("pwd");
            //check to see if user exists
            userExists = app2CA.userProfileExists(usrName);
            //if he doesn't then prompt user to fill in fields
            Application2UserData app2UserEnteredData = null;
            try
            {
                app2UserEnteredData = (Application2UserData)app2CA.getUserProfileData(usrName↙
                    , app2Name);
            }
```

```
/**
 * Here is just a class that has some properties and methods, the main thing to mention is ↙
   that
 * it implements Serializable, this is required to store data in UserProfile
 */

public class Application1UserData implements java.io.Serializable
{
    private String ssNum;
    private String password;
    private String host;
    private String additionalParam;

    public Application1UserData()
    {
        ssNum="Unknown";
        password="None";
        host="n/a";
        additionalParam="n/a";
    }

    public Application1UserData(String ssn, String pw, String ht, String ap)
    {
        ssNum = ssn;
        password = pw;
        host = ht;
        additionalParam = ap;
    }

    public void setssNum(String ssn)
    {
        ssNum = ssn;
    }

    public void setPassword(String pw)
    {
        password = pw;
    }

    public void setHost(String ht)
    {
        host = ht;
    }

    public void setAdditionalParam(String ap)
    {
        additionalParam = ap;
    }

    public String getssNum()
    {
        return ssNum;
    }

    public String getPassword()
    {
        return password;
    }

    public String getHost()
    {
        return host;
    }

    public String getAddditionalParam()
    {
        return additionalParam;
    }
}
```

```
import com.thinairapps.platform.connector.*;
import com.thinairapps.tag.html.*;
import com.thinairapps.platform.device.*;

import java.util.*;
import java.io.*;


/**
 * This Connector is called from the main menu of the portal.  It first checks to see if the ↙
   session passed in is valid
 * Then it checks to see if the user is in User Profile, if their is an entry then display  ↙
   the data, if there isn't
 * then display the configuration field and have the user enter the data.  There is a bit of ↙
   a tricky part and the way
 * I've implemented is not the most robust way.  The tricky part deals with if a user already↙
   has a User Profile, but from
 * another application.  In that case it needs to check if the data in User profile is       ↙
   associated with this application
 * in particular.  I've marked the code where the logic is taking place
 */
public class Application1Connector implements Connector
{
    ConnectorAccess app1CA;
    String app1Path;
    String app1Name;

    public void init(String appName, String appPath, Properties connectorProps,               ↙
        ConnectorAccess ca, ApplicationLog al)
    {
        app1CA = ca;
        app1Path = appPath;
        app1Name = appName;
    }

    public String[] getDevices()
    {
        return new String[] {"TA_HTML"};
    }

    public void handle(Properties appProps, Device dev, OutputStream out)
    {
        String sid;
        String usrName;
        String password;
        String result;
        String dataEntered;

        boolean valid;
        boolean userExists;
        Hashtable cache = null;

        //first get the SID, in this case the only way that a user should enter this           ↙
            application is through the menu

        sid = appProps.getProperty("sid");

        //another variable to see where the user is

        dataEntered = appProps.getProperty("dataEntered");

        //check to see if the session is valid, it might have timed out....
        valid = app1CA.sessionValid(sid);

        if (valid)
        {
            //extract user information
            //Now get the cache from the SID
            try
            {
            cache = app1CA.getSessionCache(sid);
            }
            catch (Exception e)
            {
                //catch NoSuchSessionException
            }

            //retrieve the username and password
            usrName = (String)cache.get("usr");
            password = (String)cache.get("pwd");
            //Here's the TRICKY part, now we go and look to see if there is data associated ↙
                with this
            //application in particular.  The method userProfileExists does not tell us which↙
                application
            //(if any) there is data for.
            userExists = app1CA.userProfileExists(usrName);

            Application1UserData app1UserEnteredData = null;
            try
            {
                //Here is where we try to see if there is data associated with this           ↙
                    application
                app1UserEnteredData = (Application1UserData)app1CA.getUserProfileData(usrName↙
                    , app1Name);
            }
            catch(Exception e)
            {
                //catch NoSuchUserProfileException
            }
            //Here we check 2 things
            //1)If the user does not exist, then it is his first time and display the        ↙
                configuration screen
            //2) Or he already exists, but the data is for another application, display the  ↙
                configuration screen for htis application
            if (!userExists || app1UserEnteredData == null)
            {
                //check to see if data for user has been entered
                //this needs to be passed twice, once to check if user exists, then check     ↙
                    again for entering data
                if (dataEntered != null)
                {
                //store user data
                //instatiate Applicaiton1UserData
                Application1UserData app1UserData = new Application1UserData();
                //Set the variables
                app1UserData.setssNum(appProps.getProperty("ssNum"));
                app1UserData.setPassword(appProps.getProperty("pwd"));
                app1UserData.setHost(appProps.getProperty("host"));
                app1UserData.setAdditionalParam(appProps.getProperty("additionalParam"));

                //1st create the user profile
                try
                {
                    //Here's another check, if the user already exists, and we are here that ↙
                        means
                    //there was no data for this application, but we do not need to create   ↙
                        another userprofile
                    //so if the User does not exist, then create....
                    if (!userExists)
                    {
                        app1CA.createUserProfile(usrName, password, app1Name);
                    }
                    //2nd now add the application data
                    app1CA.setUserProfileData(usrName, app1Name, app1UserData);
                }
                catch (Exception e)
                {
                    //catch exception for ProfileAlready Exists
                //catch exception for ProfileStoreLockedException
            }

            //Some HTML to tell the user that the data has been stored
            HTMLTagDocument htmlTag = new HTMLTagDocument();
            Body bodyTag = new Body();

            Paragraph pTag = new Paragraph();

            pTag.addChild(new Text("Application 1 User Added"));
            pTag.addChild(new Break());
            pTag.addChild(new Text("For User " + usrName));
            pTag.addChild(new Break());
            bodyTag.addChild(pTag);

            //View the data entered, incorporate the SID into the URL
            Anchor an1 = new Anchor("View Data", app1Path + "?sid=" + sid, new Text("View↙
                Data"));
            bodyTag.addChild(an1);
            htmlTag.addChild(bodyTag);
            try
            {
            out.write( htmlTag.render().getBytes());
            }
            catch (Exception e)
            {
                //catch out exception
            }


            }
            else
            {
                //Else the user does not exist so display the configuration screen
                result = renderInputUserDataScreen(sid);
                try
                {
                out.write(result.getBytes());
                }
                catch (Exception e)
                {
                    //catch IO excecption
                }
            }
        }
        else
        //if user exists and Application Data for this app exists that means that data  ↙
            already is there
        //so display data to user
        {
                HTMLTagDocument htmlTag = new HTMLTagDocument();
                Body bodyTag = new Body();

                Paragraph pTag = new Paragraph();

                pTag.addChild(new Text("Application 1 Data"));
                pTag.addChild(new Break());
                pTag.addChild(new Text("For User " + usrName));
                pTag.addChild(new Break());
                bodyTag.addChild(pTag);

                bodyTag.addChild(new Text("Social Number: " + app1UserEnteredData.      ↙
                    getssNum()));
                bodyTag.addChild(new Break());
                bodyTag.addChild(new Text("Password: " + app1UserEnteredData.getPassword ↙
                    ()));
                bodyTag.addChild(new Break());
                bodyTag.addChild(new Text("Host: " + app1UserEnteredData.getHost()));
                bodyTag.addChild(new Break());
            bodyTag.addChild(new Text("Additional Parameter: " + appl1UserEnteredData.↙
              getAddditionalParam()));
            bodyTag.addChild(new Break());
            bodyTag.addChild(new Text("This data is now stored in UserProfile and can↙
              be retrieved at any time with the Login and Password<br>"));
            Anchor an1 = new Anchor("Menu", "/portal?firstTime=1&sid=" + sid, new    ↙
              Text("Return to Portal Menu"));
            bodyTag.addChild(an1);
            htmlTag.addChild(bodyTag);
            try
            {
            out.write( htmlTag.render().getBytes());
            }
            catch (Exception e)
            {
              //catch out exception
            }

        }

      }
      else
      {
        String result2 = "Sorry Session is not valid or Timed Out, please login again";
        try
        {
        out.write(result2.getBytes());
        }
        catch (Exception e)
        {
        }

      }


    }

    public String renderInputUserDataScreen(String sid)
    {
      HTMLTagDocument htmlTag = new HTMLTagDocument();
      Body bodyTag = new Body();

      Paragraph pTag = new Paragraph();

      pTag.addChild(new Text("Please configure Application 1"));
      pTag.addChild(new Break());
      pTag.addChild(new Text("Enter Data"));
      pTag.addChild(new Break());
      bodyTag.addChild(pTag);

      Form formTag = new Form ("UserData", appl1Path, "POST");

      formTag.addFormElement(new LabeledInput("ssNum", "Social Security Number: "));

      formTag.addChild(new Break());

      PasswordField pwdInput = new PasswordField("pwd");

      formTag.addChild(new Text("Password: "));
      formTag.addFormElement(pwdInput);
      formTag.addChild(new Break());
      formTag.addFormElement(new LabeledInput("host", "Host: "));
      formTag.addChild(new Break());
      formTag.addFormElement(new LabeledInput("additionalParam", "Additional Parameter    ↙
        : "));
      formTag.addChild(new Break());
      formTag.addChild(new HiddenInput("sid",sid));
      formTag.addChild(new HiddenInput("dataEntered","Yes"));
      formTag.addChild(new SubmitButton ("Submit"));
```

```
        bodyTag.addChild(formTag);

        htmlTag.addChild(bodyTag);

        return htmlTag.render();

    }
}
```

```java
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWAR\E LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.provider.*;

//rendering packages used to build markup
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;
//import com.thinairapps.tag.html.*;

// the groupware packages
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.* ;

import java.util.*;
import java.io.*;

/**This sample illustrates the use of the CustomItem type to handle data in custom-created
 * folders and databases.
 * This sample renders WML. It prompts the user to choose one of two actions: add (create a ↙
   new
 * item in the specified folder), or read (get the field names and values in the first item ↙
   found
 * within that folder, and display a group of them on the screen).
 *
 * The login data (provider name, host name, username, password), the name of the template/ ↙
   form for
 * the custom item folder, the name of the folder and (for a Lotus Domino item) the name of ↙
   the
 * database, are all specified within the connector.ini file.
 *
 * For a comprehensive reference of the Groupware Library see the ThinAir Groupware javadocs.
 */
public class CRMConnector implements Connector
 {

     // The friendly name of this sample app
     protected String appName;

     //The path that needs to be appended to the server URL to access the app
     static String path;

     // Our access point to the services of ThinAir Server
     protected ConnectorAccess access;

     // The application log
     protected com.thinairapps.platform.connector.ApplicationLog log;

     // The provider
     protected String provider;

     // The user's login data
     protected String host, userName, password;

     // The location of the custom item folder within the Groupware store.
     // This should not be a global variable in a real connector.
     protected String location;

     // The name of the form/template that this custom folder uses -
```

```
    // this variable is not actually used in any of this connector's code; however,
    // it was included because it would be used by any real connector dealing with
    // CustomItems, to add new items. See the comments within the addCustomItem()
    // method for details
    static String formName = null;

    protected String sessionId = null;

    static Device g_DEVICE;

    //Action constants
    static final String DISPLAY_ACTION = "display";
    static final String ACTION_FIELD = "action";
    static final String LOGIN_ACTION = "login";
    static final String CREATE_ACTION = "add";
    static final String READ_ACTION = "read";
    static final String VIEW_ACTION = "views";
    static final String VIEW_BY_FIELD_ACTION = "view";
    static final String EDIT_ACTION = "edit";
    static final String UPDATE_ACTION = "update";

    static final String VIEW_BY_STATUS="ByStatus";
    static final String VIEW_BY_INDUSTRY="ByIndustry";
    static final String VIEW_BY_SALESCONTACT="BySalesContact";

    static final String ELEMENT_NUMBER = "elemnum";

    //By Status Constants
    static final String STS_NEEDS_FIRST_CONTACT = "NFC";
    static final String STS_NEEDS_FOLLOWUP = "NFU";
    static final String STS_NEEDS_CREDIT_APPROVAL = "NCA";
    static final String STS_NEEDS_TO_BE_INVOICED = "NTBI";
    static final String STS_CREDIT_APPROVED = "CA";
    static final String STS_INVOICE_SENT = "IS";
    static final String STS_CREDIT_DENIED = "CD";
    static final String STS_DEAD_END = "DE";

    //By Industry Constants
    static final String I_ADVERTISING = "ADV";
    static final String I_CONSULTING = "CON";
    static final String I_ENTERTAINMENT = "ENT";
    static final String I_FINANCE = "FIN";
    static final String I_GOVERNMENT = "GOV";
    static final String I_HEALTHCARE = "HEA";
    static final String I_MANUFACTURING = "MAN";
    static final String I_RETAIL = "RET";

    //By Sales Contact Constants
    static final String SC_MIKHAIL_BULGAKOV = "MB";
    static final String SC_NEIL_DIAMOND = "ND";
    static final String SC_SAM_DONALDSON = "SD";
    static final String SC_RICHARD_FEYNMAN = "RF";
    static final String SC_JOE_FRAZIER = "JF";
    static final String SC_ARTHUR_RIMBAUD = "AR";
    static final String SC_LEON_TROTSKY = "LT";
    static final String SC_MICHELLE_YEOH = "MY";


    /**
     * init() is called by the ThinAirServer when the Connector is loaded.  It provides the ↙
       connector with
     * resources it needs to interact with the ThinAirServer.
     * For more information about the Connector interface, see the javadocs for the ThinAir ↙
       Server API
     *
     * @param applicationName is a String derived from connector.ini.
     * @param applicationPath is a String derived from connector.ini.  We don't need this for↙
       this sample.
     * @param connectorProps is a Properties list containing developer assigned ↙
```

```
        connector-specific properties.
     * @param connectorAccess is our access point to the services provided by ThinAir Server.
     */
    public void init(String applicationName, String applicationPath, Properties
        connectorProps,
                    ConnectorAccess connectorAccess, com.thinairapps.platform.connector.
                        ApplicationLog appLog) throws ConnectorInitException
    {
        this.path = applicationPath;
        this.appName = applicationName;
        access = connectorAccess;
        log = appLog;

        // the two strings from connector.ini that we'll use to create the official
        // "location" string.  database is for Domino only
        String folder, database;

        // get provider name, as well as all login data, location of the custom folder, and
        // the name of the form/template being used, from the properties list (connector.ini)
        // Provider has to be either Exchange or Domino
        provider = connectorProps.getProperty("Provider");

        if (provider.length() == 0) throw new ConnectorInitException("No Provider entry in
            connector.ini");

        host = connectorProps.getProperty("Host");
        if (host.length() == 0) throw new ConnectorInitException("No Host entry in connector.
            ini");

        userName = connectorProps.getProperty("UserName");
        if (userName.length() == 0) throw new ConnectorInitException("No UserName entry in
            connector.ini");

        password = connectorProps.getProperty("Password");
        if (password.length() == 0) throw new ConnectorInitException("No Password entry in
            connector.ini");

        folder = connectorProps.getProperty("Folder");
        if (folder.length() == 0) throw new ConnectorInitException("No Folder entry in
            connector.ini");

        database = connectorProps.getProperty("Database");
        // no exception thrown if user didn't include the name of the database - this may or
            may
        // not be a necessity for the groupware store being accessed. In the case of the
            groupware
        // providers that come with the ThinAir Server, the Domino provider requires one,
            while the
        // Exchange provider doesn't

        // now, set the location string - if no database name was included, then location
        // will just be equal to the folder name
        if (database.length() == 0)
        {
            location = folder;
        }
        else
        {
            // a database name was included; since we have only a single String to represent
            // the location within the eventual data request, how do we get both the folder
            // and the database name into this one String? Thankfully, there's a utility in
            // the CustomItem class that takes care of it for us
            location = CustomItem.LocNameUtils.createLocationString(database, folder);
        }

        formName = connectorProps.getProperty("FormName");
        // no exception thrown if user didn't include the name of the folder's form/template;
        // a CustomItem connector can function without it, although not as well
    }
    /**getDevices() is called once by the ThinAir Server during start-up.  It allows a       ↙
        Connector to
     * indicate the types of devices it supports.  getDevices() returns an array containing  ↙
        the names of all
     * DeviceProfiles supported by this Connector.  These names are the friendly names used  ↙
        to uniquely
     * identify every DeviceProfile.  To get the friendly name of a particular device, refer ↙
        to the ThinAir
     * Server Developer Guide or call DeviceProfile's getName() method.
     *
     * For more details about device detection and handling see the DeviceDetective sample   ↙
        connector and the
     * ThinAir Server Developer Guide.
     *
     * @return an array of Strings representing the friendly names of the devices this       ↙
        Connector supports.
     */
    public String[] getDevices()
    {
        String devices[] = {WAPDeviceProfile.NAME,HTMLDeviceProfile.NAME,PalmVIIDeviceProfile↙
            .NAME, OmniSkyDeviceProfile.NAME};

        return devices;
    }



    /**The handle method implements the core logic of a Connector.  It takes an incoming      ↙
        request from a
     * particular device, and returns an appropriate response. This method is called whenever↙
        the server
     * receives a request from a type of device that the Connector indicates it supports,     ↙
        destined (as
     * indicated in the request URL) for a specific application. It is the responsibility of ↙
        the Connector
     * to interpret the request and generate an appropriate response.
     *
     * The server will pass a Device object containing as much information as possible into  ↙
        this method.
     * The Connector can then utilize the particular Device class to determine more detailed ↙
        information
     * on the capabilities of the particular device making the request.
     *
     * @param props a set of name value pairs corresponding to the HTTP request parameters   ↙
        from the device.
     * @param device a Device object created in the image of the actual device making this   ↙
        request.
     * @param result a reference to the OutputStream that will be returned to the device.
     */
    public void handle(Properties props, Device device, OutputStream result) throws          ↙
        IOException
    {

        String resultString = null;

        //Set the device equal to the global variable for device
        CRMConnector.g_DEVICE = device;

        //The cache for this session
        Hashtable cache = null;

        //get the 'action' parameter from the request. This is an HTTP param we define to      ↙
            determine what action
        //to take when we get a request.
        String action = props.getProperty(ACTION_FIELD);
        String view = props.getProperty(ELEMENT_NUMBER);
        try
        {
            if (action == null)
            {
                // if this is the first hit (or any request for the main deck)...
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.renderStartScreen();
                else
                    resultString = CRMHTMLRenderer.renderStartScreen();
            }
            //If all the elements on the form have been filled out, then display it
            else if (action.equals(DISPLAY_ACTION))
            {
                addCustomItem(location, sessionId, props);
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.renderMessage("The form has been        ↙
                        completed");
                else
                {
                    resultString = CRMHTMLRenderer.renderMessage("The form has been       ↙
                        completed");
                }
            }
            else if (action.equals(LOGIN_ACTION))
            {
                sessionId = loginUser(provider, host, userName, password);
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.renderOptionMenu();
                else
                    resultString = CRMHTMLRenderer.renderOptionMenu();
            }
            else
            {
                if (action.equals(CREATE_ACTION))
                {
                    if (device instanceof WAPDevice)
                        resultString = CRMWMLRenderer.renderInputForm();
                    else
                        resultString = CRMHTMLRenderer.renderInputForm();
                }
                else if (action.equals(VIEW_ACTION))
                {
                    if (device instanceof WAPDevice)
                        resultString = CRMWMLRenderer.renderAvailableViews();
                    else
                        resultString = CRMHTMLRenderer.renderAvailableViews();
                }
                //One must view an item before they edit it, so we set certain important    ↙
                    variables in
                //the view item section
                else if (action.equals(EDIT_ACTION))
                {
                    String editViewParam = props.getProperty("editPrm");

                    //From the URL, determine which element they want to see..
                    Integer prelimNumber = new Integer (0);
                    int elementNumber;
                    elementNumber = (prelimNumber.parseInt(editViewParam));

                    //Get the cache for this session
                    cache = access.getSessionCache(sessionId);

                    //Retrieve the Store Items that we've already placed into the cache
                    //A StoreItems is a Vector
                    StoreItems customItems = ((StoreItems)cache.get("storeitems"));

                    //Retrieve the particular item that we want
                    CustomItem item = ((CustomItem)(customItems.elementAt(elementNumber)));
                    //Also get the message ID
                    StoreItem storeItem = ((StoreItem)(customItems.elementAt                 ↙
                        (elementNumber)));
                    String messageID = storeItem.getID();

                    //Render the Item
                    if (device instanceof WAPDevice)
                        resultString = CRMWMLRenderer.editItem(item,messageID);
                    else
                        resultString = CRMHTMLRenderer.editItem(item,messageID);
                }
                else if (action.equals(UPDATE_ACTION))
                {
                    //Update the Item
                    updateCustomItem(location,sessionId, props);

                    //Render the message
                    if (device instanceof WAPDevice)
                        resultString = CRMWMLRenderer.renderMessage("Item has been updated");
                    else
                        resultString = CRMHTMLRenderer.renderMessage("Item has been          ↙
                            updated");
                }
                else if (action.equals(VIEW_BY_STATUS))
                {
                    StoreItems customItems = getCustomItems(location, sessionId);

                    //Get the cache for this session
                    cache = access.getSessionCache(sessionId);

                    //Put the returned StoreItems into the cache
                    cache.put("storeitems",customItems);

                    //render the fields of this object
                    if (device instanceof WAPDevice)
                        resultString = CRMWMLRenderer.renderView(customItems,                ↙
                            VIEW_BY_STATUS);
                    else
                        resultString = CRMHTMLRenderer.renderView(customItems,               ↙
                            VIEW_BY_STATUS);
                }
                else if (action.equals(VIEW_BY_INDUSTRY))
                {
                    StoreItems customItems = getCustomItems(location, sessionId);

                    //Get the cache for this session
                    cache = access.getSessionCache(sessionId);

                    //Put the returned StoreItems into the cache
                    cache.put("storeitems",customItems);

                    //render the fields of this object
                    if (device instanceof WAPDevice)
                        resultString = CRMWMLRenderer.renderView(customItems,                ↙
                            VIEW_BY_INDUSTRY);
                    else
                        resultString = CRMHTMLRenderer.renderView(customItems,               ↙
                            VIEW_BY_INDUSTRY);
                }
                else if (action.equals(VIEW_BY_SALESCONTACT))
                {
                    StoreItems customItems = getCustomItems(location, sessionId);

                    //Get the cache for this session
                    cache = access.getSessionCache(sessionId);

                    //Put the returned StoreItems into the cache
                    cache.put("storeitems",customItems);
                //render the fields of this object
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.renderView(customItems,          ↙
                        VIEW_BY_SALESCONTACT);
                else
                    resultString = CRMHTMLRenderer.renderView(customItems,         ↙
                        VIEW_BY_SALESCONTACT);
            }
            //Determine what Customer Status view the user selected
            else if (action.equals("NFC"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Needs First Contact",     ↙
                        access, sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Needs First Contact",    ↙
                        access, sessionId);
            }
            else if (action.equals("NFU"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Needs Follow-Up", access, ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Needs Follow-Up", access ↙
                        , sessionId);
            }
            else if (action.equals("NCA"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Needs Credit Approval",   ↙
                        access, sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Needs Credit Approval",  ↙
                        access, sessionId);
            }
            else if (action.equals("NTBI"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Needs to be Invoiced",    ↙
                        access, sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Needs to be Invoiced",   ↙
                        access, sessionId);
            }
            else if (action.equals("CA"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Credit Approved", access, ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Credit Approved", access ↙
                        , sessionId);
            }
            else if (action.equals("IS"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Invoice Sent", access,    ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Invoice Sent", access,   ↙
                        sessionId);
            }
            else if (action.equals("CD"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Credit Denied", access,   ↙
                        sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Credit Denied", access,  ↙
                    sessionId);
        }
        else if (action.equals("DE"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Dead End", access,        ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Dead End", access,       ↙
                    sessionId);
        }
        //Determine what Industry field the Customer selected
        else if (action.equals("ADV"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Advertising", access,     ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Advertising", access,    ↙
                    sessionId);
        }
        else if (action.equals("CON"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Consulting", access,      ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Consulting", access,     ↙
                    sessionId);
        }
        else if (action.equals("ENT"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Entertainment", access,   ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Entertainment", access,  ↙
                    sessionId);
        }
        else if (action.equals("FIN"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Finance", access,         ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Finance", access,        ↙
                    sessionId);
        }
        else if (action.equals("GOV"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Government", access,      ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Government", access,     ↙
                    sessionId);
        }
        else if (action.equals("HEA"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Healthcare", access,      ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Healthcare", access,     ↙
                    sessionId);
        }
        else if (action.equals("MAN"))
```

```
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Manufacturing", access,    ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Manufacturing", access,   ↙
                        sessionId);
            }
            else if (action.equals("RET"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Retail", access,           ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Retail", access,          ↙
                        sessionId);
            }
            //Determine what Sales Contact Field the Customer selected
            else if (action.equals("AR"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Arthur Rimbaud", access,   ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Arthur Rimbaud", access,  ↙
                        sessionId);
            }
            else if (action.equals("JF"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Joe Frazier", access,      ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Joe Frazier", access,     ↙
                        sessionId);
            }
            else if (action.equals("LT"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Leon Trotsky", access,     ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Leon Trotsky", access,    ↙
                        sessionId);
            }
            else if (action.equals("MY"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Michelle Yeoh", access,    ↙
                        sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Michelle Yeoh", access,   ↙
                        sessionId);
            }
            else if (action.equals("MB"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Mikhail Bulgakov", access  ↙
                        , sessionId);
                else
                    resultString = CRMHTMLRenderer.viewByField("Mikhail Bulgakov", access↙
                        , sessionId);
            }
            else if (action.equals("ND"))
            {
                if (device instanceof WAPDevice)
                    resultString = CRMWMLRenderer.viewByField("Neil Diamond", access,     ↙
                        sessionId);
                else
                resultString = CRMHTMLRenderer.viewByField("Neil Diamond", access,   ↙
                    sessionId);
        }
        else if (action.equals("RF"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Richard Feynman", access, ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Richard Feynman", access ↙
                    , sessionId);
        }
        else if (action.equals("SD"))
        {
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.viewByField("Sam Donaldson", access,   ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.viewByField("Sam Donaldson", access,  ↙
                    sessionId);
        }
        //Call the method that renders the fields of the selected item
        else if (action.equals(CRMConnector.VIEW_BY_FIELD_ACTION))
        {
            Integer prelimNumber = new Integer (0);
            int elementNumber;
            elementNumber = (prelimNumber.parseInt(view));

            //Get the cache for this session
            cache = access.getSessionCache(sessionId);

            //Keep a reference to the item that the user is viewing in case we want  ↙
                to edit
            //it later
            String itemKey="ItemViewed";
            cache.put(itemKey,view);

            //Retrieve the Store Items that we've already placed into the cache
            //A StoreItem is a Vector
            StoreItems customItems = ((StoreItems)cache.get("storeitems"));

            //Retrieve the particular item that we want
            CustomItem item = ((CustomItem)(customItems.elementAt(elementNumber)));

            //Render the Item
            if (device instanceof WAPDevice)
                resultString = CRMWMLRenderer.renderCustomItemFields(item, access,   ↙
                    sessionId);
            else
                resultString = CRMHTMLRenderer.renderCustomItemFields(item, access,  ↙
                    sessionId);
        }
    }
}
catch (Exception e)
{
    e.printStackTrace();
    // Here, we employ a primitive solution of simply displaying the error message  ↙
        and providing a link
    // back to the welcome screen; a larger app would handle each error separately  ↙
        and navigate the
    // user accordingly
    if (device instanceof WAPDevice)
        resultString = CRMWMLRenderer.renderMessage(e.getMessage());
    else
        resultString = CRMHTMLRenderer.renderMessage(e.getMessage());
}

// in this example we logged on, performed a simple action, then logged off again. A ↙
```

```
            more
        // complete app might hold the session open between requests, and cache the retrieved
        // StoreItems in the session cache, using this to feed item bodies out to the client
        // without going to the provider each time
        byte[] resultBytes = resultString.getBytes();
        result.write(resultBytes);
    }



    /**loginUser() logs in the user to a groupware store using the specified login data
     *
     * @param providerName the name of the provider being used to access the message store.
     * @param host the IP or server name of the message store.
     * @param userName the user name of the account being logged onto.
     * @param password the password for this user.
     * @return a providerSessionId if success; otherwise an error will be thrown.
     */
    protected String loginUser (String providerName, String host, String userName,
                                String password) throws Exception
    {
        String SID = null;

        try
        {
            // Create a new Session with the specified provider and returns a unique Session ↙
                ID.
            SID = access.createProviderSession (providerName);

            // Get the providerProxy associated with the session we just created,
            // this is what is used to interact with the Provider
            StoreProviderProxy spLite =  access.getStoreProvider (SID);

            // Create a StoreProviderLogin object, this defines the action the provider will ↙
                execute
            StoreProviderLogin login = new StoreProviderLogin (userName, password, host);

            // use the providerProxy to login. The provider returns the items it supports
            SupportedItems supports = spLite.connectUser (login);

            // check to make sure that this provider handles CustomItem objects
            boolean supportsCustItems = false;
            Enumeration supportedEnum = supports.getItems();
            while (supportedEnum.hasMoreElements())
            {
                SupportedItem curItem = (SupportedItem)supportedEnum.nextElement();
                if (curItem.getType() == ItemTypes.CUSTOM_ITEM)
                    supportsCustItems = true;
            }

            // if it doesn't handle CustomItem objects, throw an exception
            if (!supportsCustItems)
                throw new Exception("Specified provider (" + providerName + ") does not      ↙
                    support CustomItem handling");

        }
        catch (NoSuchProviderException e)
        {
            throw new Exception("No Provider named " + providerName + " was loaded by the    ↙
                ThinAir Server");
        }
        return SID;
    }


    /**getCustomItems() retrieves items from a groupware location, and returns a
     * CustomItems object containing all its information
     *
     * @param location The location in the groupware store being accessed
     * @param SID The session ID for the user's connection to the groupware store
     * @return a CustomItem representing the first item in the folder
     */
    protected StoreItems getCustomItems(String location, String SID) throws Exception
    {
        String resultString;

        ItemRequest iReq = new ItemRequest ();
        iReq.itemType = ItemTypes.CUSTOM_ITEM;
        iReq.itemLocation = location;

        //Use -1 to specify no maximum limit
        iReq.max = -1;
        iReq.startID = null;
        iReq.bounds = null;

        UserDataRequest udReq = new UserDataRequest ();
        udReq.requests = new ItemRequest[1];
        udReq.requests[0] = iReq;

        UserData uData = access.getStoreProvider(SID).getUserData(udReq);
        ItemRequestResponse irr = uData.responses[0];
        StoreItems customItems = irr.items;

        return customItems;
    }



    /**addCustomItem() adds an item to a groupware location containing custom-
     * defined items
     *
     * @param location The location in the groupware store being accessed
     * @param SID The session ID for the user's connection to the groupware store
     */
    protected void addCustomItem (String location, String SID, Properties props) throws
        Exception
    {
    try {
        //Create a custom item
        //String formName = "IPM.Contact.Sample";
        //String formName = "Sample";
        //From Connector.ini
        //String formName = connectorProps.getProperty("FormName");
        //String providerName = "Domino";

        CustomItem custItem = new CustomItem(ItemTypes.CONTACT, formName);

        //Example of how to set the standard fields of a custom Item
        //Each CustomItem has a StandardItem included within it
        //Contact testContact = new Contact();
        //custItem.setStandardItem (testContact);
        //testContact.setFullName("Rudy Guliani");

        //Set the time
        Calendar cal = Calendar.getInstance();
        cal.setTime(new Date());
        Date todayTime = cal.getTime();
        String timeString = todayTime.toString();
        custItem.addField("ItemCreated", timeString);

        //If its a Domino Provider, then set the "Form" field to the form name
        //Future iterations of the Domino Provider will do this automatically
        if (provider.equals("Domino"))
        {
            custItem.addField("Form", formName);
        }
        //Set the fields of that item
        custItem.addField("CustomerName", props.getProperty("cstnm"));
        custItem.addField("Position", props.getProperty("psn"));
        custItem.addField("CompanyName", props.getProperty("cnm"));

        //Industry
        String industry = props.getProperty("industry");

        String[] industryURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
        String[] industries                                                    ↙
            = {"Advertising","Consulting","Entertainment","Finance","Government","Healthcare"↙
            ,"Manufacturing","Retail"};
        int i;
        for (i = 0; i < 8; i++)
        {
            if (industry.equals(industryURLParams[i]))
            {
                custItem.addField("Industry", industries[i]);
                break;
            }
        }

        //Sales Contact
        String salesContact = props.getProperty("sc");
        String[] salesContactURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
        String[] salesContacts = {"Mikhail Bulgakov","Neil Diamond","Sam Donaldson","Richard ↙
            Feynman","Joe Frazier","Arthur Rimbaud","Leon Trotsky","Michelle Yeoh"};

        for (i = 0; i < 8; i++)
        {
            if (salesContact.equals(salesContactURLParams[i]))
            {
                custItem.addField("SalesContact", salesContacts[i]);
                break;
            }
        }

        //Account Number
        custItem.addField("AccountNumber", props.getProperty("an"));

        //Customer Status
        String customerStatus = props.getProperty("custstatus");
        String[] customerStatusURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
        String[] customerStatusVals = {"Needs First Contact","Needs Follow-Up","Needs Credit ↙
            Approval","Needs to be Invoiced","Credit Approved","Invoice Sent","Credit       ↙
            Denied","Dead End"};

        for (i = 0; i < 8; i++)
        {
            if (customerStatus.equals(customerStatusURLParams[i]))
            {
                custItem.addField("CustomerStatus", customerStatusVals[i]);
                break;
            }
        }

        // set the location of the new item to be the user-specified location
        custItem.setLocationInStore(location);

        StoreProviderProxy spProxy = access.getStoreProvider (SID);

        AddNewGroupwareItem addAction = new AddNewGroupwareItem(custItem);

        spProxy.doUserDataAction (addAction);
        return;
    }
        catch (Exception e)
        {
            System.out.println ("Error adding the item: " + e);
        }
    }




    /**addCustomItem() adds an item to a groupware location containing custom-
     * defined items
     *
     * @param location The location in the groupware store being accessed
     * @param SID The session ID for the user's connection to the groupware store
     * @param props The properties object containing the request
     */
    protected void updateCustomItem(String location, String SID, Properties props) throws
        Exception
    {
        try
        {
        //Create a custom item
        //String formName = "IPM.Contact.Sample";
        //From Connector.ini
        String formName = props.getProperty("FormName");

        //Get the messageID
        String messageID = props.getProperty("MessageID");

        //Create the new CustomItem object.  We're using it only as a container.
        CustomItem custItemContainer = new CustomItem(ItemTypes.CONTACT, formName);


        //Set the fields of that item
        custItemContainer.addField("CustomerName", props.getProperty("cstnm"));
        custItemContainer.addField("Position", props.getProperty("psn"));
        custItemContainer.addField("CompanyName", props.getProperty("cnm"));

        //Industry
        String industry = props.getProperty("industry");

        String[] industryURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
        String[] industries
            = {"Advertising","Consulting","Entertainment","Finance","Government","Healthcare"
            ,"Manufacturing","Retail"};
        int i;
        for (i = 0; i < 8; i++)
        {
            if (industry.equals(industryURLParams[i]))
            {
                custItemContainer.addField("Industry", industries[i]);
                break;
            }
        }

        //Sales Contact
        String salesContact = props.getProperty("sc");
        String[] salesContactURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
        String[] salesContacts = {"Mikhail Bulgakov","Neil Diamond","Sam Donaldson","Richard
            Feynman","Joe Frazier","Arthur Rimbaud","Leon Trotsky","Michelle Yeoh"};

        for (i = 0; i < 8; i++)
        {
            if (salesContact.equals(salesContactURLParams[i]))
            {
                custItemContainer.addField("SalesContact", salesContacts[i]);
                break;
            }
        }
        //Account Number
        custItemContainer.addField("AccountNumber", props.getProperty("an"));

        //Customer Status
        String customerStatus = props.getProperty("custstatus");

        String[] customerStatusURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
        String[] customerStatusVals = {"Needs First Contact","Needs Follow-Up","Needs Credit ↙
            Approval","Needs to be Invoiced","Credit Approved","Invoice Sent","Credit       ↙
            Denied","Dead End"};

        for (i = 0; i < 8; i++)
        {
            if (customerStatus.equals(customerStatusURLParams[i]))
            {
                custItemContainer.addField("CustomerStatus", customerStatusVals[i]);
                break;
            }
        }

            //New Contact object - this will be CustomItem's standard item
            Contact actualContact = new Contact();
            custItemContainer.setStandardItem (actualContact);

            // set the location of the new item to be the user-specified location
            custItemContainer.setLocationInStore(location);

            StoreProviderProxy spProxy = access.getStoreProvider (SID);

            //AddNewGroupwareItem addAction = new AddNewGroupwareItem(custItem);
            //spProxy.doUserDataAction (addAction);
            spProxy.doUserDataAction (new UpdateGroupwareItem(messageID, location,          ↙
                custItemContainer));
        }
        catch (Exception e)
        {
            System.out.println ("Error updating the item");
        }
        return;
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.provider.*;

//ThinAir Tag Libraries imports
import com.thinairapps.tag.*;
import com.thinairapps.tag.html.*;

// the groupware packages
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.*;

//Standard Java imports
import java.util.*;



/**
 * This utility class renders output as HTML for a variety of devices
 */
class CRMHTMLRenderer
{
    /**This method renders a deck containing a welcome card
     *
     * @return the rendered deck.
     */
    static String renderStartScreen()
    {
        HTMLTagDocument doc = new HTMLTagDocument();
        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
        if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE
            instanceof OmniSkyDevice)
            {
            Meta meta = new Meta("name", "PalmComputingPlatform", "true");
            head.addChild(meta);
            }
        com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM
            Connector");
        head.addChild(bold);
        doc.setHead(head);

        Body body = new Body();

        com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
        body.addChild(para);

        String href = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +
            CRMConnector.LOGIN_ACTION;
        com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Start",
            href, new com.thinairapps.tag.html.Text("Login"));

        body.addChild(an1);
        body.addChild(new com.thinairapps.tag.html.Break());

        doc.setBody(body);

        String resultString = doc.render();
        return resultString;
    }



    /**
     * This method renders a deck with a card that lets the user specify which action to take
     *
     * @return the rendered deck.
     */
    static String renderOptionMenu()
    {
        HTMLTagDocument doc = new HTMLTagDocument();
        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
        if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE      ↙
            instanceof OmniSkyDevice)
            {
            Meta meta = new Meta("name", "PalmComputingPlatform", "true");
            head.addChild(meta);
            }
        com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM   ↙
            Connector");
        head.addChild(bold);
        doc.setHead(head);

        Body mBody = new Body();

        com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
        para.addChild(new com.thinairapps.tag.html.Text("Please choose from the following    ↙
            ..."));

        mBody.addChild(para);

        String href = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +            ↙
            CRMConnector.CREATE_ACTION + "&amp;rnd=" + Math.random();
        com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Create",   ↙
            href, new com.thinairapps.tag.html.Text("Create a new Item"));

        mBody.addChild(an1);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        String href2 = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +           ↙
            CRMConnector.VIEW_ACTION + "&amp;rnd=" + Math.random();
        com.thinairapps.tag.html.Anchor an2 = new com.thinairapps.tag.html.Anchor("Select",   ↙
            href2, new com.thinairapps.tag.html.Text("Select a View"));

        mBody.addChild(an2);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        doc.setBody(mBody);

        String resultString = doc.render();
        return resultString;
    }


    /**
     * Renders the fields of a single CustomItem
     *
     * @param item the CustomItem whose fields we should render
     * @return the rendered deck.
     */
    static String renderCustomItemFields(CustomItem item, ConnectorAccess access, String     ↙
        sessionId) throws Exception
    {
        //Get the cache for this session
        Hashtable cache=null;
        cache = access.getSessionCache(sessionId);
        //Storing the item being viewed in the cache
        String itemViewed;
        itemViewed = cache.get("ItemViewed").toString();

        //Get the messageID
        String messageID = ((StoreItem)item).getID();

        //create the deck
        String url = null;

        HTMLTagDocument doc = new HTMLTagDocument();
        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
        if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE          ↙
            instanceof OmniSkyDevice)
            {
            Meta meta = new Meta("name", "PalmComputingPlatform", "true");
            head.addChild(meta);
            }
        com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM    ↙
            Connector");
        head.addChild(bold);
        doc.setHead(head);

        Body mBody = new Body();

        com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
        para.addChild(new com.thinairapps.tag.html.Text("Item Fields:"));

        mBody.addChild(para);

        // Now add the fields and their values:

        //first get the Data object that contains all the info about our custom fields.
        Data customFields = item.getCustomFieldData();

        //Get an enumeration of the fields...
        Enumeration fieldEnum = customFields.getFields();

        // go through the fields, and add each one to the deck - we'll stop after 15,
        // to avoid any deck overflow problems
        int itemsDisplayed = 0;
        while (fieldEnum.hasMoreElements() && itemsDisplayed < 15)
        {
            String fieldTextName = null;
            String fieldTextValue = null;
            Field thisField = (Field)fieldEnum.nextElement();

            //We must check the type, because that determines how we retrieve the field
            if (thisField.getType() == Field.BOOLEAN_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getBoolean();
            }
            else if (thisField.getType() == Field.DOUBLE_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getDouble();
            }
            else if (thisField.getType() == Field.INT_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getInt();
            }
            else if (thisField.getType() == Field.LONG_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getLong();
            }
            else if (thisField.getType() == Field.STRING_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getString();
            }
            else if (thisField.getType() == Field.DATE_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getDate();
            }
            //Transform the field name to the what we want to display as the field name
            //We do this because we want the actual field names in Domino or Exchange to have
            //no spaces, but we want the display names to have spaces (i.e.                  ↙
                CustomerName--Customer Name)
            String fieldDisplay=null;
            if (fieldTextName.equals ("CustomerName"))
                fieldDisplay="Customer Name";
            else if (fieldTextName.equals ("Position"))
                fieldDisplay="Position";
            else if (fieldTextName.equals ("CompanyName"))
                fieldDisplay="Company Name";
            else if (fieldTextName.equals ("Industry"))
                fieldDisplay="Industry";
            else if (fieldTextName.equals ("ItemCreated"))
                fieldDisplay="Item Created";
            else if (fieldTextName.equals ("SalesContact"))
                fieldDisplay="Sales Contact";
            else if (fieldTextName.equals ("AccountNumber"))
                fieldDisplay="Account Number";
            else if (fieldTextName.equals ("CustomerStatus"))
                fieldDisplay="Customer Status";
            //If there'a a field on the form that we're not expecting,
            //don't display it
            else
            {
                continue;
            }

            mBody.addChild(new Text(fieldDisplay));
            mBody.addChild(new Text(fieldTextValue));

            mBody.addChild(new Break());
            itemsDisplayed++;
        }

        //Link Home
        String href = CRMConnector.path+ "?rnd="+Math.random();
        com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Home",       ↙
            href, new com.thinairapps.tag.html.Text("Start Again..."));

        mBody.addChild(new com.thinairapps.tag.html.Break());
        mBody.addChild(an1);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        //Link to edit the Item
        String editHref = CRMConnector.path+"?"+ CRMConnector.ACTION_FIELD + "=" +              ↙
            CRMConnector.EDIT_ACTION + "&amp;editPrm=" + itemViewed + "&amp;rnd="+Math.random↙
            ();
        com.thinairapps.tag.html.Anchor an2 = new com.thinairapps.tag.html.Anchor("Edit",       ↙
            editHref, new com.thinairapps.tag.html.Text("Edit Item"));

        mBody.addChild(an2);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        doc.setBody(mBody);

        String resultString = doc.render();
        return resultString;
    }
```

```
/** This method renders a simple message, either an error or a success,
  * then links back to the main page
  * @param message the message to be presented to the user
  * @return the rendered HTML deck
  */
 static String renderMessage (String message)
 {
     HTMLTagDocument doc = new HTMLTagDocument();
     com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
     if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE       ↙
         instanceof OmniSkyDevice)
         {
         Meta meta = new Meta("name", "PalmComputingPlatform", "true");
         head.addChild(meta);
         }
     com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM   ↙
         Connector");
     head.addChild(bold);
     doc.setHead(head);

     Body body = new Body();

     com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
     para.addChild(new com.thinairapps.tag.html.Text(message));
     body.addChild(para);
     body.addChild(new com.thinairapps.tag.html.Break());

     //Link home
     String href = CRMConnector.path+ "?rnd="+Math.random();

     com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Start",   ↙
         href, new com.thinairapps.tag.html.Text("Start again..."));

     body.addChild(an1);
     body.addChild(new com.thinairapps.tag.html.Break());

     doc.setBody(body);

     String resultString = doc.render();
     return resultString;
 }



 /**
  * This method renders a deck with several cards including a welcome card and card for   ↙
     entering information
  *
  * This method makes use of the ThinAir HTML Tag Library for HTML markup creation.  For   ↙
     more information
  * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
     Development Guide.
  *
  * @return the rendered deck.
  */
 static String renderInputForm()
 {
     HTMLTagDocument doc = new HTMLTagDocument();
     com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
     if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE       ↙
         instanceof OmniSkyDevice)
         {
         Meta meta = new Meta("name", "PalmComputingPlatform", "true");
         head.addChild(meta);
         }
```

```
        com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM    ↙
            Connector");
        head.addChild(bold);
        doc.setHead(head);

        Body body = new Body();

        com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
        body.addChild(para);

        //Create the form
        String href="/crm";
        com.thinairapps.tag.html.Form inputForm = new com.thinairapps.tag.html.Form ("Sample ↙
            Form",href,"POST");

        //Create the inputs and selects
        com.thinairapps.tag.html.LabeledInput custName = new LabeledInput ("cstnm","Customer ↙
            Name:");
        com.thinairapps.tag.html.LabeledInput psnName = new LabeledInput ("psn","Position:");
        com.thinairapps.tag.html.LabeledInput compName = new LabeledInput ("cnm","Company     ↙
            Name:");
        com.thinairapps.tag.html.Select industryName = new Select ("industry");
            industryName.addOption ("a", "Advertising");
            industryName.addOption ("b", "Consulting");
            industryName.addOption ("c", "Entertainment");
            industryName.addOption ("d", "Finance");
            industryName.addOption ("e", "Government");
            industryName.addOption ("f", "Health Care");
            industryName.addOption ("g", "Manufacturing");
            industryName.addOption ("h", "Retail");
        com.thinairapps.tag.html.Select salesContactName = new Select ("sc");
            salesContactName.addOption ("a","Mikhail Bulgakov");
            salesContactName.addOption ("b","Neil Diamond");
            salesContactName.addOption ("c","Sam Donaldson");
            salesContactName.addOption ("d","Richard Feynman");
            salesContactName.addOption ("e","Joe Frazier");
            salesContactName.addOption ("f","Arthur Rimbaud");
            salesContactName.addOption ("g","Leon Trotsky");
            salesContactName.addOption ("h","Michelle Yeoh");
        com.thinairapps.tag.html.LabeledInput anName = new LabeledInput ("an","Account Number↙
            :");
        com.thinairapps.tag.html.Select custstatusName = new Select ("custstatus");
            custstatusName.addOption ("a","Needs First Contact");
            custstatusName.addOption ("b","Needs Follow-Up");
            custstatusName.addOption ("c","Needs Credit Approval");
            custstatusName.addOption ("d","Needs to be Invoiced");
            custstatusName.addOption ("e","Credit Approved");
            custstatusName.addOption ("f","Invoice Sent");
            custstatusName.addOption ("g","Credit Denied");
            custstatusName.addOption ("h","Dead End");
        com.thinairapps.tag.html.SubmitButton submit = new SubmitButton ("Submit","Submit");

        //Add the inputs and selects to the Form

        //Hidden
        inputForm.addChild(new Input("hidden", "action", "display"));

        inputForm.addChild (new LabeledInput ("cstnm","Customer Name:"));
        inputForm.addChild (new com.thinairapps.tag.html.Break());

        inputForm.addChild (new LabeledInput ("psn","Position:"));
        inputForm.addChild (new com.thinairapps.tag.html.Break());

        inputForm.addChild (new LabeledInput ("cnm","Company Name:"));
        inputForm.addChild (new com.thinairapps.tag.html.Break());

        inputForm.addChild (new Text("Industry:&nbsp;"));
        inputForm.addChild (industryName);
        inputForm.addChild (new com.thinairapps.tag.html.Break());
        inputForm.addChild (new Text("Sales Contact:&nbsp;"));
        inputForm.addChild (salesContactName);
        inputForm.addChild (new com.thinairapps.tag.html.Break());

        inputForm.addChild (new LabeledInput ("an","Account Number:"));
        inputForm.addChild (new com.thinairapps.tag.html.Break());

        inputForm.addChild (new Text("Customer Status:&nbsp;"));
        inputForm.addChild (custstatusName);
        inputForm.addChild (new com.thinairapps.tag.html.Break());
        inputForm.addChild (new com.thinairapps.tag.html.Break());

        inputForm.addChild (submit);
        inputForm.addChild (new com.thinairapps.tag.html.Break());



        //Set the href with the values from the user
        //href = CRMConnector.path + "?action=display&amp;cstnm=$cstnm&amp;psn=$psn&amp;cnm=$↙
            cnm&amp;industry=$industry&amp;spm=$spm&amp;sc=$salesContact&amp;an=$an&amp;    ↙
            custstatus=$custstatus&amp;rnd="+Math.random();
        //Add the Form to the body
        body.addChild(inputForm);
        body.addChild(new com.thinairapps.tag.html.Break());
        //Add the body to the document
        doc.setBody(body);

        String resultString = doc.render();
        return resultString;
    }



    /**
     * Display to the user the available ways that they can view the data in the folder.
     * renderOptionMenu() renders an option screen, users select one of those options and
     * Handle() checks the URL and then calls this method if users wanted to see all the     ↙
        views
     * available to them
     *
     * @return the rendered HTML deck
     */
    static String renderAvailableViews ()
    {
        HTMLTagDocument doc = new HTMLTagDocument();
        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
        if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE          ↙
            instanceof OmniSkyDevice)
            {
            Meta meta = new Meta("name", "PalmComputingPlatform", "true");
            head.addChild(meta);
            }
        com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM    ↙
            Connector");
        head.addChild(bold);
        doc.setHead(head);

        Body mBody = new Body();

        com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
        para.addChild(new com.thinairapps.tag.html.Text("Select a View:"));

        mBody.addChild(para);

        String href = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +           ↙
            CRMConnector.VIEW_BY_STATUS + "&amp;rnd=" + Math.random();
        com.thinairapps.tag.html.Anchor an1 = new com.thinairapps.tag.html.Anchor("Status",  ↙
            href, new com.thinairapps.tag.html.Text("View by Status"));

        mBody.addChild(an1);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        String href2 = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +
            CRMConnector.VIEW_BY_INDUSTRY + "&amp;rnd=" + Math.random();
        com.thinairapps.tag.html.Anchor an2 = new com.thinairapps.tag.html.Anchor
            ("Industry", href2, new com.thinairapps.tag.html.Text("View by Industry"));

        mBody.addChild(an2);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        String href3 = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +
            CRMConnector.VIEW_BY_SALESCONTACT + "&amp;rnd=" + Math.random();
        com.thinairapps.tag.html.Anchor an3 = new com.thinairapps.tag.html.Anchor
            ("SalesContact", href3, new com.thinairapps.tag.html.Text("View by Sales
            Contact"));

        mBody.addChild(an3);
        mBody.addChild(new com.thinairapps.tag.html.Break());

        doc.setBody(mBody);

        String resultString = doc.render();
        return resultString;
    }



    /**
     * This method renders the fields in a selected view.
     * TO DO -- display the number of items that have each field
     * TO DO -- turn this connector into a collection of widgets that
     * are more generic
     *
     * @param customItems All of the items in the folder
     * @param view The view that the user wants to use on these items
     * @return A rendered HTML deck
     */
    static String renderView (StoreItems customItems, String view)
    {
        HTMLTagDocument doc = new HTMLTagDocument();
        com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
        if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE
            instanceof OmniSkyDevice)
            {
            Meta meta = new Meta("name", "PalmComputingPlatform", "true");
            head.addChild(meta);
            }
        com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM
            Connector");
        head.addChild(bold);
        doc.setHead(head);

        Body mBody = new Body();
        com.thinairapps.tag.html.Paragraph p = new com.thinairapps.tag.html.Paragraph();

        if (view.equals (CRMConnector.VIEW_BY_STATUS))
        {
            p.addChild(new Text("View By Customer Status:"));
            p.addChild(new Break());

            //add the Paragraph
            mBody.addChild(p);
            com.thinairapps.tag.html.Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT);

            /**
            //Links to the possible actions
            String nfcHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.STS_NEEDS_FIRST_CONTACT + "&amp;rnd=" + Math.random();
            String nfHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +       ↙
                CRMConnector.STS_NEEDS_FOLLOWUP + "&amp;rnd=" + Math.random();
            String ncaHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.STS_NEEDS_CREDIT_APPROVAL + "&amp;rnd=" + Math.random();
            String ntbiHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +     ↙
                CRMConnector.STS_NEEDS_TO_BE_INVOICED + "&amp;rnd=" + Math.random();
            String caHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +       ↙
                CRMConnector.STS_CREDIT_APPROVED + "&amp;rnd=" + Math.random();
            String isHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +       ↙
                CRMConnector.STS_INVOICE_SENT + "&amp;rnd=" + Math.random();
            String cdHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +       ↙
                CRMConnector.STS_CREDIT_DENIED + "&amp;rnd=" + Math.random();
            String deHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +       ↙
                CRMConnector.STS_DEAD_END + "&amp;rnd=" + Math.random();


            com.thinairapps.tag.html.Anchor nfcAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("FirstContact", nfcHref, new com.thinairapps.tag.html.Text("Needs First     ↙
                Contact"));
            com.thinairapps.tag.html.Anchor nfAnchor = new com.thinairapps.tag.html.Anchor   ↙
                ("FollowUp", nfHref, new com.thinairapps.tag.html.Text("Needs Follow-Up"));
            com.thinairapps.tag.html.Anchor ncaAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("CreditApproval", ncaHref, new com.thinairapps.tag.html.Text("Needs Credit  ↙
                Approval"));
            com.thinairapps.tag.html.Anchor ntbiAnchor = new com.thinairapps.tag.html.Anchor ↙
                ("Invoice", ntbiHref, new com.thinairapps.tag.html.Text("Needs to be         ↙
                Invoiced"));
            com.thinairapps.tag.html.Anchor caAnchor = new com.thinairapps.tag.html.Anchor   ↙
                ("CreditApproved", caHref, new com.thinairapps.tag.html.Text("Credit         ↙
                Approved"));
            com.thinairapps.tag.html.Anchor isAnchor = new com.thinairapps.tag.html.Anchor   ↙
                ("InvoiceSent", isHref, new com.thinairapps.tag.html.Text("Invoice Sent"));
            com.thinairapps.tag.html.Anchor cdAnchor = new com.thinairapps.tag.html.Anchor   ↙
                ("CreditDenied", cdHref, new com.thinairapps.tag.html.Text("Credit Denied"));
            com.thinairapps.tag.html.Anchor deAnchor = new com.thinairapps.tag.html.Anchor   ↙
                ("DeadEnd", deHref, new com.thinairapps.tag.html.Text("Dead End"));

            mBody.addChild(nfcAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(nfAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(ncaAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(ntbiAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(caAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(isAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(cdAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(deAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            */
//////////////////////////////////////////////////////////////////
            //Add links to all the possible actions
            String [] statusHREFNames                                                        ↙
                = {"nfcHref","nfHref","ncaHref","ntbiHref","caHref","isHref","cdHref","deHref↙
                "};
            String [] statusURLParams = {CRMConnector.STS_NEEDS_FIRST_CONTACT,CRMConnector.  ↙
                STS_NEEDS_FOLLOWUP,CRMConnector.STS_NEEDS_CREDIT_APPROVAL,CRMConnector.      ↙
                STS_NEEDS_TO_BE_INVOICED,CRMConnector.STS_CREDIT_APPROVED,CRMConnector.      ↙
                STS_INVOICE_SENT,CRMConnector.STS_CREDIT_DENIED,CRMConnector.STS_DEAD_END};
            String [] statusAnchorValues                                                     ↙
                = {"FirstContact","FollowUp","CreditApproval","Invoice","CreditApproved","Inv↙
                oiceSent","CreditDenied","DeadEnd"};
            String [] statusAnchorDisplayText = {"Needs First Contact","Needs                ↙
```

```
                Follow-Up","Needs Credit Approval","Needs to be Invoiced","Credit      ↙
                Approved","Invoice Sent","Credit Denied","Dead End"};
            com.thinairapps.tag.html.Anchor [] statusAnchorNames = {new com.thinairapps.tag. ↙
                html.Anchor("nfcAnchor"),new com.thinairapps.tag.html.Anchor("nfAnchor"),new ↙
                com.thinairapps.tag.html.Anchor("ncaAnchor"),new com.thinairapps.tag.html.   ↙
                Anchor("ntbiAnchor"),new com.thinairapps.tag.html.Anchor("caAnchor"),new com.↙
                thinairapps.tag.html.Anchor("isAnchor"),new com.thinairapps.tag.html.Anchor  ↙
                ("cdAnchor"),new com.thinairapps.tag.html.Anchor("deAnchor")};

            int i;
            for (i = 0; i < 8; i++)
            {
                statusHREFNames[i] = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD      ↙
                    + "=" + statusURLParams[i]+ "&amp;rnd=" + Math.random();
                statusAnchorNames[i] = new com.thinairapps.tag.html.Anchor(statusAnchorValues↙
                    [i],statusHREFNames[i],new com.thinairapps.tag.html.Text                 ↙
                    (statusAnchorDisplayText[i]));
                mBody.addChild(statusAnchorNames[i]);
                mBody.addChild(new com.thinairapps.tag.html.Break());
            }
//////////////////////////////////////////////////////////////////
            //add the body
            doc.setBody(mBody);
        }
        else if (view.equals (CRMConnector.VIEW_BY_INDUSTRY))
        {
            p.addChild(new Text("View By Industry Name:"));
            p.addChild(new Break());

            //add the Paragraph
            mBody.addChild(p);

            com.thinairapps.tag.html.Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT);

            /**
            //Links to the possible actions
            String advHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_ADVERTISING + "&amp;rnd=" + Math.random();
            String conHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_CONSULTING + "&amp;rnd=" + Math.random();
            String entHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_ENTERTAINMENT + "&amp;rnd=" + Math.random();
            String finHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_FINANCE + "&amp;rnd=" + Math.random();
            String govHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_GOVERNMENT + "&amp;rnd=" + Math.random();
            String heaHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_HEALTHCARE + "&amp;rnd=" + Math.random();
            String manHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_MANUFACTURING + "&amp;rnd=" + Math.random();
            String retHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                CRMConnector.I_RETAIL + "&amp;rnd=" + Math.random();

            com.thinairapps.tag.html.Anchor advAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Advertising", advHref, new com.thinairapps.tag.html.Text("Advertising"));
            com.thinairapps.tag.html.Anchor conAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Consulting", conHref, new com.thinairapps.tag.html.Text("Consulting"));
            com.thinairapps.tag.html.Anchor entAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Entertainment", entHref, new com.thinairapps.tag.html.Text                ↙
                ("Entertainment"));
            com.thinairapps.tag.html.Anchor finAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Finance", finHref, new com.thinairapps.tag.html.Text("Finance"));
            com.thinairapps.tag.html.Anchor govAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Government", govHref, new com.thinairapps.tag.html.Text("Government"));
            com.thinairapps.tag.html.Anchor heaAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Healthcare", heaHref, new com.thinairapps.tag.html.Text("Healthcare"));
            com.thinairapps.tag.html.Anchor manAnchor = new com.thinairapps.tag.html.Anchor  ↙
                ("Manufacturing", manHref, new com.thinairapps.tag.html.Text                ↙
                ("Manufacturing"));
            com.thinairapps.tag.html.Anchor retAnchor = new com.thinairapps.tag.html.Anchor ↙
                ("Retail", retHref, new com.thinairapps.tag.html.Text("Retail"));

            mBody.addChild(advAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(conAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(entAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(finAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(govAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(heaAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(manAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(retAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            */
//////////////////////////////////////////////////////////////////
            //Add links to all the possible actions
            String [] industryHREFNames                                                  ↙
                = {"advHref","conHref","entHref","finHref","govHref","heaHref","manHref","ret↙
                Href"};
            String [] industryURLParams = {CRMConnector.I_ADVERTISING,CRMConnector.           ↙
                I_CONSULTING,CRMConnector.I_ENTERTAINMENT,CRMConnector.I_FINANCE,CRMConnector↙
                .I_GOVERNMENT,CRMConnector.I_HEALTHCARE,CRMConnector.I_MANUFACTURING,         ↙
                CRMConnector.I_RETAIL};
            String [] industryAnchorValues                                               ↙
                = {"Advertising","Consulting","Entertainment","Finance","Government","Healthc↙
                are","Manufacturing","Retail"};
            String [] industryAnchorDisplayText                                          ↙
                = {"Advertising","Consulting","Entertainment","Finance","Government","Healthc↙
                are","Manufacturing","Retail"};
            com.thinairapps.tag.html.Anchor [] industryAnchorNames = {new com.thinairapps.tag↙
                .html.Anchor("advAnchor"),new com.thinairapps.tag.html.Anchor("conAnchor"),  ↙
                new com.thinairapps.tag.html.Anchor("entAnchor"),new com.thinairapps.tag.html↙
                .Anchor("finHref"),new com.thinairapps.tag.html.Anchor("govAnchor"),new com. ↙
                thinairapps.tag.html.Anchor("heaAnchor"),new com.thinairapps.tag.html.Anchor ↙
                ("manAnchor"),new com.thinairapps.tag.html.Anchor("retAnchor")};

            int i;
            for (i = 0; i < 8; i++)
            {
                industryHREFNames[i] = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD    ↙
                    + "=" + industryURLParams[i]+ "&amp;rnd=" + Math.random();
                industryAnchorNames[i] = new com.thinairapps.tag.html.Anchor                 ↙
                    (industryAnchorValues[i],industryHREFNames[i],new com.thinairapps.tag.   ↙
                    html.Text(industryAnchorDisplayText[i]));
                mBody.addChild(industryAnchorNames[i]);
                mBody.addChild(new com.thinairapps.tag.html.Break());
            }
//////////////////////////////////////////////////////////////////
            //add the body
            doc.setBody(mBody);
        }
        else if (view.equals (CRMConnector.VIEW_BY_SALESCONTACT))
        {
            p.addChild(new Text("View By Sales Contact:"));
            p.addChild(new Break());

            //add the Paragraph
            mBody.addChild(p);

            com.thinairapps.tag.html.Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT);


            //Links to the possible actions
            String arHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_ARTHUR_RIMBAUD + "&amp;rnd=" + Math.random();
            String jfHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_JOE_FRAZIER + "&amp;rnd=" + Math.random();
            String ltHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_LEON_TROTSKY + "&amp;rnd=" + Math.random();
            String myHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_MICHELLE_YEOH + "&amp;rnd=" + Math.random();
            String mbHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_MIKHAIL_BULGAKOV + "&amp;rnd=" + Math.random();
            String ndHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_NEIL_DIAMOND + "&amp;rnd=" + Math.random();
            String rfHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_RICHARD_FEYNMAN + "&amp;rnd=" + Math.random();
            String sdHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
                CRMConnector.SC_SAM_DONALDSON + "&amp;rnd=" + Math.random();

            com.thinairapps.tag.html.Anchor arAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Rimbaud", arHref, new com.thinairapps.tag.html.Text("Arthur Rimbaud"));
            com.thinairapps.tag.html.Anchor jfAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Frazier", jfHref, new com.thinairapps.tag.html.Text("Joe Frazier"));
            com.thinairapps.tag.html.Anchor ltAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Trotsky", ltHref, new com.thinairapps.tag.html.Text("Leon Trotsky"));
            com.thinairapps.tag.html.Anchor myAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Yeoh", myHref, new com.thinairapps.tag.html.Text("Michelle Yeoh"));
            com.thinairapps.tag.html.Anchor mbAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Bulgakov", mbHref, new com.thinairapps.tag.html.Text("Mikhail Bulgakov"));
            com.thinairapps.tag.html.Anchor ndAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Diamond", ndHref, new com.thinairapps.tag.html.Text("Neil Diamond"));
            com.thinairapps.tag.html.Anchor rfAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Feynman", rfHref, new com.thinairapps.tag.html.Text("Richard Feynman"));
            com.thinairapps.tag.html.Anchor sdAnchor = new com.thinairapps.tag.html.Anchor    ↙
                ("Donaldson", sdHref, new com.thinairapps.tag.html.Text("Sam Donaldson"));

            mBody.addChild(arAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(jfAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(ltAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(myAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(mbAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(ndAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(rfAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
            mBody.addChild(sdAnchor);
            mBody.addChild(new com.thinairapps.tag.html.Break());
///////////////////////////////////////////////////////////////////
            /**
            //Add links to all the possible actions
            String [] salescontactHREFNames                                                   ↙
                = {"arHref","jfHref","ltHref","myHref","mbHref","ndHref","rfHref","sdHref"};
            String [] salescontactURLParams = {CRMConnector.SC_ARTHUR_RIMBAUD,CRMConnector.   ↙
                SC_JOE_FRAZIER,CRMConnector.SC_LEON_TROTSKY,CRMConnector.SC_MICHELLE_YEOH,    ↙
                CRMConnector.SC_MIKHAIL_BULGAKOV,CRMConnector.SC_NEIL_DIAMOND,CRMConnector.   ↙
                SC_RICHARD_FEYNMAN,CRMConnector.SC_SAM_DONALDSON};
            String [] salescontactAnchorValues                                                ↙
                = {"Rimbaud","Frazier","Trotsky","Yeoh","Bulgakov","Diamond","Feynman","Donal↙
                dson"};
            String [] salescontactAnchorDisplayText = {"Arthur Rimbaud","Joe Frazier","Leon   ↙
                Trotsky","Michelle Yeoh","Mikhail Bulgakov","Neil Diamond","Richard           ↙
                Feynman","Sam Donaldson"};
            com.thinairapps.tag.html.Anchor [] salescontactAnchorNames = {new com.thinairapps↙
                .tag.html.Anchor("arAnchor"),new com.thinairapps.tag.html.Anchor("jfAnchor"),↙
                new com.thinairapps.tag.html.Anchor("ltAnchor"),new com.thinairapps.tag.html.↙
                Anchor("myAnchor"),new com.thinairapps.tag.html.Anchor("mbAnchor"),new com.  ↙
            thinairapps.tag.html.Anchor("ndAnchor"),new com.thinairapps.tag.html.Anchor ↙
            ("rfAnchor"),new com.thinairapps.tag.html.Anchor("sdAnchor")};

        int i;
        for (i = 0; i < 8; i++)
        {
            salescontactHREFNames[i] = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD↙
                + "=" + salescontactURLParams[i]+ "&amp;rnd=" + Math.random();
            salescontactAnchorNames[i] = new com.thinairapps.tag.html.Anchor             ↙
                (salescontactAnchorValues[i],salescontactHREFNames[i],new com.thinairapps↙
                .tag.html.Text(salescontactAnchorDisplayText[i]));
            mBody.addChild(salescontactAnchorNames[i]);
            mBody.addChild(new com.thinairapps.tag.html.Break());
        }
        */
///////////////////////////////////////////////////////////////////////

        //add the body
        doc.setBody(mBody);
    }

    String resultString = doc.render();
    return resultString;
}


/**
 *A user selects a field value by which to sort the contents of the folder and
 * then this method is called to display all the items that have that value
 *
 * @param fieldValue The value of the field by which the user wants to sort the folder
 * @param access A handle to ConnectorAccess and the ThinAir Server services
 * @param sessionId An identifier of the user's already established session
 * @return A collection of StoreItems that satisfy the criteria of the user
 */
static String viewByField(String fieldValue, ConnectorAccess access, String sessionId)   ↙
    throws Exception
{
    HTMLTagDocument doc = new HTMLTagDocument();
    com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
    if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE        ↙
        instanceof OmniSkyDevice)
        {
        Meta meta = new Meta("name", "PalmComputingPlatform", "true");
        head.addChild(meta);
        }
    com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM   ↙
        Connector");
    head.addChild(bold);
    doc.setHead(head);

    Body mBody = new Body();

    com.thinairapps.tag.html.Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT);

    p.addChild(new Text ("Matching Items:"));
    p.addChild(new Break());

    //add the Paragraph
    mBody.addChild(p);
    com.thinairapps.tag.html.Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT);

    //The cache for this session
    Hashtable cache = null;

    //The Vector that will store the items that have fields that match the incoming field↙
        parameter
    Vector itemswMatchingfields = new Vector(15);
```

```
        //Get the cache for this session
        cache = access.getSessionCache(sessionId);

        //Retieve the Store Items that we've already placed into the cache
        StoreItems customItems = ((StoreItems)cache.get("storeitems"));

        //we get an Enumeration of the items...
        Enumeration sortedItemEnum = (((Vector)customItems).elements());

        boolean didAnyItemsMatch = false;

        //Go through the items, and identify those that have the field that
        //has been passed in as the search parameter.
        int itemIterated = 0;
        String elementNumber = "elemnum";
        String href = "";
        com.thinairapps.tag.html.Anchor itemAnchor = new com.thinairapps.tag.html.Anchor("", ↙
            href);
        while (sortedItemEnum.hasMoreElements())
        {
            String fieldText = null;
            String companyName = null;
            CustomItem custItem = (CustomItem)sortedItemEnum.nextElement();

            //Get the fields of the item
            Data customFields = custItem.getCustomFieldData();

            //Get an enumeration of the fields
            Enumeration fieldEnum = customFields.getFields();

            //If the search parameter matches a field on the item, then we return a link to ↙
                the
            //item with the item's company name displayed.
            boolean hasField = false;
            while (fieldEnum.hasMoreElements())
            {
                Field thisField = (Field)fieldEnum.nextElement();
                //Get the Item's Company Name field.  We need it for displaying a link to the↙
                    item.
                if (thisField.getName().equals ("CompanyName"))
                {
                    companyName = thisField.getString();
                    if (hasField == true)
                    {
                        //Create the link to the Item, with the Company Name field rendered
                        href = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +     ↙
                            CRMConnector.VIEW_BY_FIELD_ACTION + "&amp;elemnum=" +             ↙
                            itemIterated + "&amp;rnd=" + Math.random();

                        //Initialize the anchor
                        itemAnchor = new Anchor ("CompanyName",href,new com.thinairapps.tag. ↙
                            html.Text(companyName));
                        //Add the Break
                        mBody.addChild(new com.thinairapps.tag.html.Break());
                        //Add the anchor
                        mBody.addChild(itemAnchor);

                        didAnyItemsMatch = true;

                        //Add to our Vector of items with matching fields
                        itemswMatchingfields.addElement(custItem);
                        companyName = null;
                        hasField = false;
                        break;
                    }
                    else
                        continue;
```

```
      }

      //We will display links only to those items that match the criteria that the
      //user is asking for.
      else if (thisField.getString().equals(fieldValue))
      {
        hasField = true;
        if (!(companyName == null))
        {
          //Create the link to the Item, with the Company Name field rendered
          href = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +        ↙
            CRMConnector.VIEW_BY_FIELD_ACTION + "&amp;elemnum=" +                  ↙
            itemIterated + "&amp;rnd=" + Math.random();

          //Initialize the anchor
          itemAnchor = new Anchor ("CompanyName",href,new com.thinairapps.tag.  ↙
            html.Text(companyName));
          //Add the Break
          mBody.addChild(new com.thinairapps.tag.html.Break());
          //Add the anchor
          mBody.addChild(itemAnchor);

          didAnyItemsMatch = true;

          //Add to our Vector of items with matching fields
          itemswMatchingfields.addElement(custItem);
          companyName = null;
          hasField = false;
          break;
        }
        else
          continue;
      }
    } // end while
  itemIterated++;
  }


  //If no items matched the criteria, render this fact
  if (didAnyItemsMatch == false)
  {
    mBody.addChild(new com.thinairapps.tag.html.Break());
    mBody.addChild(new Text("No Items to Display"));
    mBody.addChild(new com.thinairapps.tag.html.Break());
    mBody.addChild(new com.thinairapps.tag.html.Break());
  }

  else if (!(didAnyItemsMatch == false))
  {
    mBody.addChild(new com.thinairapps.tag.html.Break());
  }
  //link home.
  String startHref = CRMConnector.path+ "?rnd="+Math.random();

  //Initialize the anchor
  com.thinairapps.tag.html.Anchor startAnchor = new Anchor ("Start",startHref,new com. ↙
    thinairapps.tag.html.Text("Start again..."));
  //Add the Break
  mBody.addChild(new com.thinairapps.tag.html.Break());
  //Add the anchor
  mBody.addChild(startAnchor);

  //add the body
  doc.setBody(mBody);

  return doc.render();
}
```

```
    /**
     * Renders an input form with the values preset
     *
     * Pass in the field values that the item had.
     *
     */
    static String editItem(CustomItem item, String messageID)
    {
    //Get the Data object that contains all our custom fields.
    Data customFields = item.getCustomFieldData();

    //Get the fields that we're expecting
    String customerNameField = customFields.getField("CustomerName").valueToString();
    String positionField = customFields.getField("Position").valueToString();
    String companyNameField = customFields.getField("CompanyName").valueToString();
    String industryField = customFields.getField("Industry").valueToString();
    String itemCreatedField = customFields.getField("ItemCreated").valueToString();
    String salesContactField = customFields.getField("SalesContact").valueToString();
    String accountNumberField = customFields.getField("AccountNumber").valueToString();
    String customerStatusField = customFields.getField("CustomerStatus").valueToString();


    HTMLTagDocument doc = new HTMLTagDocument();
    com.thinairapps.tag.html.Head head = new com.thinairapps.tag.html.Head();
    if (CRMConnector.g_DEVICE instanceof PalmVIIDevice || CRMConnector.g_DEVICE instanceof  ↙
        OmniSkyDevice)
        {
        Meta meta = new Meta("name", "PalmComputingPlatform", "true");
        head.addChild(meta);
        }
    com.thinairapps.tag.html.Bold bold = new com.thinairapps.tag.html.Bold("Sample CRM  ↙
        Connector");
    head.addChild(bold);
    doc.setHead(head);

    Body body = new Body();

    com.thinairapps.tag.html.Paragraph para = new com.thinairapps.tag.html.Paragraph();
    body.addChild(para);

    //Create the form
    String href="/crm";
    com.thinairapps.tag.html.Form inputForm = new com.thinairapps.tag.html.Form ("Sample  ↙
        Form",href,"POST");

    //Create the inputs and selects
    com.thinairapps.tag.html.LabeledInput custName = new LabeledInput ("cstnm","input",  ↙
        customerNameField,"Customer Name:");
    com.thinairapps.tag.html.LabeledInput psnName = new LabeledInput ("psn","input",  ↙
        positionField,"Position:");
    com.thinairapps.tag.html.LabeledInput compName = new LabeledInput ("cnm","input",  ↙
        companyNameField,"Company Name:");



    //Industry
    com.thinairapps.tag.html.Select industryName = new Select ("industry");
    String[] industryURLParams  = {"a", "b", "c", "d", "e","f","g","h"};
    String[] industries                                                              ↙
        = {"Advertising","Consulting","Entertainment","Finance","Government","Healthcare","Ma↙
        nufacturing","Retail"};
    com.thinairapps.tag.html.Option[] industryOptions=new Option [8];
    int i;
    for (i = 0; i < 8; i++)
    {
        industryOptions[i]= new Option (industryURLParams[i],industries[i]);
        if (industryField.equals(industries[i]))
        industryOptions[i].setSelected(true);
    else
    {
        industryOptions[i].setSelected(false);
    }
    industryName.addOption(industryOptions[i]);
}


//Sales Contact
com.thinairapps.tag.html.Select salesContactName = new Select ("sc");
String[] salesContactURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
String[] salesContacts = {"Mikhail Bulgakov","Neil Diamond","Sam Donaldson","Richard      ↙
    Feynman","Joe Frazier","Arthur Rimbaud","Leon Trotsky","Michelle Yeoh"};
com.thinairapps.tag.html.Option [] salesContactOptions = new Option[8];
int j;
for (j = 0; j < 8; j++)
{
        salesContactOptions[j]=new Option (salesContactURLParams[j],salesContacts[j]);
        if (salesContactField.equals (salesContacts[j]))
            salesContactOptions[j].setSelected(true);
        else
        {
            salesContactOptions[j].setSelected(false);
        }
        salesContactName.addOption(salesContactOptions[j]);
}


//Customer Status
com.thinairapps.tag.html.Select custstatusName = new Select ("custstatus");
String[] custstatusURLParams   = {"a", "b", "c", "d", "e","f","g","h"};
String[] custstati = {"Needs First Contact","Needs Follow-Up","Needs Credit          ↙
    Approval","Needs to be Invoiced","Credit Approved","Invoice Sent","Credit        ↙
    Denied","Dead End"};
com.thinairapps.tag.html.Option [] custstatusOptions = new Option[8];
int h;
for (h = 0; h < 8; h++)
{
        custstatusOptions[h]=new Option (custstatusURLParams[h],custstati[h]);
        if (customerStatusField.equals (custstati[h]))
            custstatusOptions[h].setSelected(true);
        else
        {
            custstatusOptions[h].setSelected(false);
        }
        custstatusName.addOption(custstatusOptions[h]);
}

com.thinairapps.tag.html.LabeledInput anName = new LabeledInput ("an","input",      ↙
    accountNumberField,"Account Number:");

com.thinairapps.tag.html.SubmitButton submit = new SubmitButton ("Submit","Submit");

//Add the inputs and selects to the Form

//Hidden
inputForm.addChild(new Input("hidden", "action", "update"));
inputForm.addChild(new Input("hidden", "MessageID", messageID));

inputForm.addChild (anName);
inputForm.addChild (new com.thinairapps.tag.html.Break());

inputForm.addChild (compName);
inputForm.addChild (new com.thinairapps.tag.html.Break());

inputForm.addChild (custName);
inputForm.addChild (new com.thinairapps.tag.html.Break());
```

```
    inputForm.addChild (new Text("Customer Status:&nbsp;"));
    inputForm.addChild (custstatusName);
    inputForm.addChild (new com.thinairapps.tag.html.Break());

    inputForm.addChild (new Text("Industry:&nbsp;"));
    inputForm.addChild (industryName);
    inputForm.addChild (new com.thinairapps.tag.html.Break());

    inputForm.addChild (psnName);
    inputForm.addChild (new com.thinairapps.tag.html.Break());

    inputForm.addChild (new Text("Sales Contact:&nbsp;"));
    inputForm.addChild (salesContactName);
    inputForm.addChild (new com.thinairapps.tag.html.Break());
    inputForm.addChild (new com.thinairapps.tag.html.Break());

    inputForm.addChild (submit);
    inputForm.addChild (new com.thinairapps.tag.html.Break());

    //Add the Form to the body
    body.addChild(inputForm);
    body.addChild(new com.thinairapps.tag.html.Break());
    //Add the body to the document
    doc.setBody(body);

    String resultString = doc.render();
    return resultString;
    }
}
```

```
/** ACCESS TO AND USE OF THIS SOFTWARE IS GOVERNED BY THE TERMS OF A SOFTWARE
 *  LICENSE AGREEMENT BETWEEN THINAIRAPPS, INC. AND LICENSEE. ANY ACCESS OR
 *  USE OF THE SOFTWARE IN VIOLATION OF THE SOFTWARE LICENSE AGREEMENT IS
 *  STRICTLY PROHIBITED.
 */

//core ThinAir Server API functionality
import com.thinairapps.platform.*;
import com.thinairapps.platform.device.*;
import com.thinairapps.platform.connector.*;
import com.thinairapps.platform.exception.*;
import com.thinairapps.platform.provider.*;

//ThinAir Tag Libraries imports
import com.thinairapps.tag.*;
import com.thinairapps.tag.wml.*;

// the groupware packages
import thinairapps.groupware.api.*;
import thinairapps.groupware.api.actions.*;
import thinairapps.groupware.api.bounds.*;
import thinairapps.groupware.api.exception.*;

//Standard Java imports
import java.util.*;



/**
 * This utility class renders output as WML for a variety of devices
 */
class CRMWMLRenderer
{
    /**This method renders a deck containing a welcome card
     *
     * @return the rendered deck.
     */
    static String renderStartScreen()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create a card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_NOWRAP);
        Bold b = new Bold(new Text("Sample CRM Connector"));
        p.addChild(b);
        p.addChild(new Break());

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        //Link
        String loginHref = CRMConnector.path + "?" + CRMConnector.ACTION_FIELD + "=" +
            CRMConnector.LOGIN_ACTION + "&amp;rnd=" + Math.random();

        //Go task for the href
        Go loginGo = new Go(loginHref,true,Go.METHOD_GET);

        Anchor loginAnchor;

        //Anchor for the Go task
        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
        {
            p.addChild(new Break());
            p.addChild(new Break());
            loginAnchor = new Anchor(loginGo,new Text("Login"));
            p.addChild(loginAnchor);
        }
        else
        {
            loginAnchor = new Anchor(loginGo,new Text("Login"));
            p.addChild(loginAnchor);
        }

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the card to the deck
        deck.addCard(card1);

        String resultString = deck.render();

        return resultString;
    }


    /**
     * This method renders a deck with a card that lets the user specify which action to take
     *
     * @return the rendered deck.
     */
    static String renderOptionMenu()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create a card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_NOWRAP);

        Bold b = new Bold(new Text("Sample CRM Connector"));
        p.addChild(b);
        p.addChild(new Break());

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        // links to the two possible actions
        String createHref =CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +
            CRMConnector.CREATE_ACTION + "&amp;rnd=" + Math.random();
        String readHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +
            CRMConnector.VIEW_ACTION + "&amp;rnd=" + Math.random();

        // Go tasks for the two hrefs
        Go createGo = new Go(createHref,true,Go.METHOD_GET);
        Go readGo = new Go(readHref,true,Go.METHOD_GET);

        Anchor createAnchor;
        Anchor readAnchor;

        //Anchors for the two Go tasks
        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
        {
            p.addChild(new Break());
            createAnchor = new Anchor(createGo,new Text("Create a new item"));
            p.addChild(createAnchor);
            p.addChild(new Break());
            readAnchor = new Anchor(readGo,new Text("Select View"));
            p.addChild(readAnchor);
            p.addChild(new Break());
        }
        else
        {
            createAnchor = new Anchor(createGo,new Text("Create a new item"));
            p.addChild(createAnchor);
            readAnchor = new Anchor(readGo,new Text("Select View"));
            p.addChild(readAnchor);
        }

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the card to the deck
        deck.addCard(card1);

        String resultString = deck.render();

        return resultString;
    }


    /**
     * Renders the fields of a single CustomItem
     *
     * @param item the CustomItem whose fields we should render
     * @return the rendered deck.
     */
    static String renderCustomItemFields(CustomItem item, ConnectorAccess access, String
        sessionId) throws Exception
    {
        //Get the cache for this session
        Hashtable cache=null;
        cache = access.getSessionCache(sessionId);

        //Storing the item being viewed in the cache
        String itemViewed;
        itemViewed = cache.get("ItemViewed").toString();

        //Get the messageID
        String messageID = ((StoreItem)item).getID();

        //create the deck
        WMLTagDocument deck = new WMLTagDocument();
        String url = null;

        //create the first card in the deck and give it the ID 'c1'
        DisplayCard card = new DisplayCard("c1");

        Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);
        Bold b = new Bold(new Text("Item Fields:"));
        p.addChild(b);
        p.addChild(new Break());
        card.addParagraph (p);
        Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        // Now add the fields and their values:

        //first get the Data object that contains all the info about our custom fields.
        Data customFields = item.getCustomFieldData();

        //Get an enumeration of the fields...
        Enumeration fieldEnum = customFields.getFields();

        // go through the fields, and add each one to the deck - we'll stop after 15,
        // to avoid any deck overflow problems
        int itemsDisplayed = 0;
        while (fieldEnum.hasMoreElements() && itemsDisplayed < 15)
        {
            String fieldTextName = null;
            String fieldTextValue = null;
            Field thisField = (Field)fieldEnum.nextElement();

            //We must check the type, because that determines how we retrieve the field
            if (thisField.getType() == Field.BOOLEAN_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getBoolean();
            }
            else if (thisField.getType() == Field.DOUBLE_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getDouble();
            }
            else if (thisField.getType() == Field.INT_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getInt();
            }
            else if (thisField.getType() == Field.LONG_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getLong();
            }
            else if (thisField.getType() == Field.STRING_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getString();
            }
            else if (thisField.getType() == Field.DATE_VAL)
            {
                fieldTextName = thisField.getName();
                fieldTextValue = ": " + thisField.getDate();
            }

            //Transform the field name to the what we want to display as the field name
            //We do this because we want the actual field names in Domino or Exchange to have
            //no spaces, but we want the display names to have spaces (i.e.              ↙
                CustomerName--Customer Name)
            String fieldDisplay=null;
            if (fieldTextName.equals ("CustomerName"))
                fieldDisplay="Customer Name";
            else if (fieldTextName.equals ("Position"))
                fieldDisplay="Position";
            else if (fieldTextName.equals ("CompanyName"))
                fieldDisplay="Company Name";
            else if (fieldTextName.equals ("Industry"))
                fieldDisplay="Industry";
            else if (fieldTextName.equals ("ItemCreated"))
                fieldDisplay="Item Created";
            else if (fieldTextName.equals ("SalesContact"))
                fieldDisplay="Sales Contact";
            else if (fieldTextName.equals ("AccountNumber"))
                fieldDisplay="Account Number";
            else if (fieldTextName.equals ("CustomerStatus"))
                fieldDisplay="Customer Status";

            //If there'a a field on the form that we're not expecting,
            //don't display it
            else
            {
                continue;
            }
            p2.addChild(new Text(fieldDisplay));
            p2.addChild(new Text(fieldTextValue));
            p2.addChild(new Break());
            itemsDisplayed++;
        }

        // link home.
        String href = CRMConnector.path+ "?rnd="+Math.random();

        //Edit link
        //One parameter must be dedicated to insuring that we have the same item to edit
        String editHref = CRMConnector.path+"?"+ CRMConnector.ACTION_FIELD + "=" +       ↙
            CRMConnector.EDIT_ACTION + "&amp;editPrm=" + itemViewed + "&amp;rnd="+Math.random↙
            ();

        Go go = new Go(href,true,Go.METHOD_GET);
        Go editGo = new Go(editHref, true, Go.METHOD_GET);

        Anchor anchor;
        Anchor editAnchor;

        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
        {
            p2.addChild(new Break());
            anchor = new Anchor(go,new Text("Start again..."));
            p2.addChild(anchor);
            editAnchor = new Anchor(editGo,new Text("Edit Item"));
            p2.addChild(editAnchor);
        }
        else
        {
            anchor = new Anchor(go,new Text("Start again..."));
            p2.addChild(anchor);
            editAnchor = new Anchor(editGo,new Text("Edit Item"));
            p2.addChild(editAnchor);
        }

        card.addParagraph(p2);
        deck.addCard (card);
        return deck.render();
    }



    /** This method renders a simple message, either an error or a success,
     * then links back to the main page
     * @param message the message to be presented to the user
     * @return the rendered WML deck
     */
    static String renderMessage (String message)
    {
        WMLTagDocument deck = new WMLTagDocument();
        DisplayCard card = new DisplayCard();
        Paragraph p = new Paragraph();

        p.addChild(new Text(message));
        p.addChild(new Break());

        //Link home
        String href = CRMConnector.path+ "?rnd="+Math.random();

        Go go = new Go(href,true,Go.METHOD_GET);

        Anchor anchor;
        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
        {
            p.addChild(new Break());
            anchor = new Anchor(go,new Text("Start again..."));
            p.addChild(anchor);
        }
        else
        {
            anchor = new Anchor(go,new Text("Start again..."));
            p.addChild(anchor);
        }

        card.addParagraph(p);
        deck.addCard(card);

        String resultString = deck.render();

        return resultString;
    }



    /**
     * This method renders a deck with several cards including a welcome card and card for   ↙
       entering information
     *
     * This method makes use of the ThinAir WML Tag Library for WML markup creation.  For     ↙
       more information
     * on use of the Tag Libraries, see the Tag Library documentation and the ThinAir Server ↙
       Development Guide.
     *
     * @return the rendered deck.
     */
    static String renderInputForm()
    {
        //Create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //Create a MultipleInputCard
        MultipleInputCard card1 = new MultipleInputCard("g1");

        //Allow the user to type in information and set the input text to lowercase by        ↙
            default
        LabeledInput custName = new LabeledInput("cstnm","Customer Name:");
        custName.setFormat("*m");

        LabeledInput position = new LabeledInput("psn","Position:");
        position.setFormat("*m");

        LabeledInput compName = new LabeledInput("cnm","Company Name:");
        compName.setFormat("*m");

        LabeledInput[] inputs1 = {custName, position, compName};

        //A link to the second card
        card1.buildCard("#g2","OK",inputs1, Go.METHOD_GET);
        deck.addCard(card1);

        //Create a select card
        SelectInputCard card2 = new SelectInputCard("g2");

        Option advertising = new Option("OK","a","Advertising");
        Option consulting = new Option("OK","b","Consulting");
        Option entertainment = new Option("OK","c","Entertainment");
        Option finance = new Option ("OK","d","Finance");
        Option government = new Option ("OK","e","Government");
        Option healthcare = new Option ("OK","f","Health Care");
        Option manufacturing = new Option("OK","g","Manufacturing");
        Option retail = new Option("OK","h","Retail");

        //Make an array of all the options
        Option[] options1 = {advertising, consulting, entertainment, finance, government,     ↙
            healthcare, manufacturing, retail};

        //Build the card.
        card2.buildCard("#g3","Industry:","industry",options1,Paragraph.ALIGN_LEFT,Paragraph.↙
            MODE_NOWRAP);

        //Add the SelectInputCard
        deck.addCard(card2);

        //Create another Select card
        SelectInputCard card5 = new SelectInputCard ("g3");
        Option bulgakov = new Option("OK","a","Mikhail Bulgakov");
        Option diamond = new Option("OK","b","Neil Diamond");
        Option donaldson = new Option("OK","c","Sam Donaldson");
        Option feynman = new Option("OK","d","Richard Feynman");
        Option frazier = new Option("OK","e","Joe Frazier");
        Option rimbaud = new Option("OK","f","Arthur Rimbaud");
        Option trotsky = new Option("OK","g","Leon Trotsky");
        Option yeoh = new Option("OK","h","Michelle Yeoh");

        Option[] options5 = {bulgakov, diamond, donaldson, feynman, frazier, rimbaud, trotsky↙
            , yeoh};

        //Build the card
        card5.buildCard("#g4","Sales Contact:","salesContact",options5,Paragraph.ALIGN_LEFT, ↙
            Paragraph.MODE_NOWRAP);

        //Add the SelectInputCard
        deck.addCard(card5);

        //Create a MultipleInputCard
        MultipleInputCard card3 = new MultipleInputCard("g4");

        LabeledInput actNumber = new LabeledInput("an","Account Number:");
        actNumber.setFormat("*m");

        LabeledInput[] inputs2 = {actNumber};

        card3.buildCard("#g5","OK",inputs2, Go.METHOD_GET);
        deck.addCard(card3);

        //Create a select card
        SelectInputCard card4 = new SelectInputCard("g5");

        Option needsfirstcontact = new Option("OK","a","Needs First Contact");
        Option needsfollowup = new Option("OK","b","Needs Follow-Up");
        Option needscreditapproval = new Option("OK","c","Needs Credit Approval");
        Option needstobeinvoiced = new Option ("OK","d","Needs to be Invoiced");
        Option creditapproved = new Option ("OK","e","Credit Approved");
        Option invoicesent = new Option ("OK","f","Invoice Sent");
        Option creditdenied = new Option("OK","g","Credit Denied");
        Option deadend = new Option("OK","h","Dead End");

        //Make an array of all the options
        Option[] options2 = {needsfirstcontact, needsfollowup, needscreditapproval,        ↙
            needstobeinvoiced, creditapproved, invoicesent, creditdenied, deadend};

        //Set the URL params to the values in the WML variables
        //&amp;, the escape sequence for ampersand, delimits name-value pairs. $ is used to  ↙
            dereference a WML variable.
        String href;
        href = CRMConnector.path + "?action=display&amp;cstnm=$cstnm&amp;psn=$psn&amp;cnm=$  ↙
            cnm&amp;industry=$industry&amp;spm=$spm&amp;sc=$salesContact&amp;an=$an&amp;      ↙
            custstatus=$custstatus&amp;rnd="+Math.random();


        //Build the card.
        card4.buildCard(href,"Customer Status:","custstatus",options2,Paragraph.ALIGN_LEFT,  ↙
            Paragraph.MODE_NOWRAP);

        //Add the SelectInputCard
        deck.addCard(card4);
        //Render the deck
        return deck.render();
    }



    /**
     * Display to the user the available ways that they can view the data in the folder.
     * renderOptionMenu() renders an option screen, users select one of those options and
     * Handle() checks the URL and then calls this method if users wanted to see all the  ↙
        views
     * available to them
     *
     * @return the rendered WML deck
     */
    static String renderAvailableViews ()
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create a card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_NOWRAP);
        Bold b = new Bold(new Text("Select a View"));
        p.addChild(b);
        p.addChild(new Break());

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        // links to the possible actions
        String byStatusHref =CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +  ↙
            CRMConnector.VIEW_BY_STATUS + "&amp;rnd=" + Math.random();
        String byIndustryHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +  ↙
            CRMConnector.VIEW_BY_INDUSTRY + "&amp;rnd=" + Math.random();
        String bySalesContactHref = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD  ↙
            + "=" + CRMConnector.VIEW_BY_SALESCONTACT + "&amp;rnd=" + Math.random();

        // Go tasks for the hrefs
        Go byStatusGo = new Go(byStatusHref,true,Go.METHOD_GET);
        Go byIndustryGo = new Go(byIndustryHref,true,Go.METHOD_GET);
        Go bySalesContactGo = new Go(bySalesContactHref,true,Go.METHOD_GET);

        Anchor byStatusAnchor;
        Anchor byIndustryAnchor;
        Anchor bySalesContactAnchor;

        // Anchors for the two Go tasks
        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
        {
            p.addChild(new Break());
            byStatusAnchor = new Anchor(byStatusGo,new Text("View by Status"));
            p.addChild(byStatusAnchor);

            p.addChild(new Break());
            byIndustryAnchor = new Anchor(byIndustryGo,new Text("View by Industry Name"));
            p.addChild(byIndustryAnchor);

            p.addChild(new Break());
            bySalesContactAnchor = new Anchor(bySalesContactGo,new Text("View by Sales  ↙
                Contact"));
            p.addChild(bySalesContactAnchor);

            p.addChild(new Break());
        }
        else
        {
            byStatusAnchor = new Anchor(byStatusGo,new Text("View by Status"));
            p.addChild(byStatusAnchor);

            byIndustryAnchor = new Anchor(byIndustryGo,new Text("View by Industry Name"));
            p.addChild(byIndustryAnchor);

            bySalesContactAnchor = new Anchor(bySalesContactGo,new Text("View by Sales         ↙
                Contact"));
            p.addChild(bySalesContactAnchor);
        }

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the card to the deck
        deck.addCard(card1);

        String resultString = deck.render();

        return resultString;
    }


    /**
     * This method renders the fields in a selected view.
     * TO DO -- display the number of items that have each field
     * TO DO -- turn this connector into a collection of widgets that
     * are more generic
     *
     * @param customItems All of the items in the folder
     * @param view The view that the user wants to use on these items
     * @return A rendered WML deck
     */
    static String renderView (StoreItems customItems, String view)
    {
        //create the deck
        WMLTagDocument deck = new WMLTagDocument();

        //create a card in the deck and give it the ID 'c1'
        DisplayCard card1 = new DisplayCard("c1");

        //create a centered Paragraph
        Paragraph p = new Paragraph(Paragraph.ALIGN_CENTER,Paragraph.MODE_NOWRAP);

        if (view.equals (CRMConnector.VIEW_BY_STATUS))
        {
            Bold b = new Bold(new Text("View By Customer Status:"));
            p.addChild(b);
            p.addChild(new Break());

            //add the Paragraph to the card
            card1.addParagraph(p);

            p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

            //Add links to all the possible actions
            String [] statusHREFNames                                                          ↙
                = {"nfcHref","nfHref","ncaHref","ntbiHref","caHref","isHref","cdHref","deHref↙
                "};
            String [] statusURLParams = {CRMConnector.STS_NEEDS_FIRST_CONTACT,CRMConnector.   ↙
                STS_NEEDS_FOLLOWUP,CRMConnector.STS_NEEDS_CREDIT_APPROVAL,CRMConnector.        ↙
                STS_NEEDS_TO_BE_INVOICED,CRMConnector.STS_CREDIT_APPROVED,CRMConnector.        ↙
                STS_INVOICE_SENT,CRMConnector.STS_CREDIT_DENIED,CRMConnector.STS_DEAD_END};
            Go[] statusGoNames = {new Go(statusHREFNames[0],true,Go.METHOD_GET),new Go        ↙
                (statusHREFNames[1],true,Go.METHOD_GET),new Go(statusHREFNames[2],true,Go.     ↙
                METHOD_GET),new Go(statusHREFNames[3],true,Go.METHOD_GET),new Go               ↙
            (statusHREFNames[4],true,Go.METHOD_GET),new Go(statusHREFNames[5],true,Go.    ↙
            METHOD_GET),new Go(statusHREFNames[6],true,Go.METHOD_GET),new Go              ↙
            (statusHREFNames[7],true,Go.METHOD_GET)};
        String [] statusAnchorDisplayText = {"Needs First Contact","Needs                 ↙
            Follow-Up","Needs Credit Approval","Needs to be Invoiced","Credit             ↙
            Approved","Invoice Sent","Credit Denied","Dead End"};
        com.thinairapps.tag.wml.Anchor [] statusAnchorNames ={new com.thinairapps.tag.wml↙
            .Anchor(statusGoNames[0],new Text(statusAnchorDisplayText[0])),new com.       ↙
            thinairapps.tag.wml.Anchor(statusGoNames[1],new Text(statusAnchorDisplayText  ↙
            [1])),new com.thinairapps.tag.wml.Anchor(statusGoNames[2],new Text            ↙
            (statusAnchorDisplayText[2])),new com.thinairapps.tag.wml.Anchor              ↙
            (statusGoNames[3],new Text(statusAnchorDisplayText[3])),new com.thinairapps.  ↙
            tag.wml.Anchor(statusGoNames[4],new Text(statusAnchorDisplayText[4])),new com↙
            .thinairapps.tag.wml.Anchor(statusGoNames[5],new Text(statusAnchorDisplayText↙
            [5])),new com.thinairapps.tag.wml.Anchor(statusGoNames[6],new Text            ↙
            (statusAnchorDisplayText[6])),new com.thinairapps.tag.wml.Anchor              ↙
            (statusGoNames[7],new Text(statusAnchorDisplayText[7]))};
        int i;
        for (i = 0; i < 8; i++)
        {
            statusHREFNames[i] = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD       ↙
                + "=" + statusURLParams[i]+ "&amp;rnd=" + Math.random();
            statusGoNames[i] = new Go(statusHREFNames[i],true,Go.METHOD_GET);
            if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
            {
                p.addChild(new Break());
                statusAnchorNames[i] = new Anchor(statusGoNames[i], new Text              ↙
                    (statusAnchorDisplayText[i]));
                p.addChild(statusAnchorNames[i]);
                p.addChild(new Break());
            }
            else
            {
                statusAnchorNames[i] = new Anchor(statusGoNames[i], new Text              ↙
                    (statusAnchorDisplayText[i]));
                p.addChild(statusAnchorNames[i]);
            }
        }

        //add the second Paragraph to the card
        card1.addParagraph(p);

        //add the card to the deck
        deck.addCard(card1);
    }
    else if (view.equals (CRMConnector.VIEW_BY_INDUSTRY))
    {
        Bold b = new Bold(new Text("View By Industry Name:"));
        p.addChild(b);
        p.addChild(new Break());

        //add the Paragraph to the card
        card1.addParagraph(p);

        p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        //Add links to all the possible actions
        String [] industryHREFNames                                                       ↙
            = {"advHref","conHref","entHref","finHref","govHref","heaHref","manHref","ret↙
            Href"};
        String [] industryURLParams = {CRMConnector.I_ADVERTISING,CRMConnector.           ↙
            I_CONSULTING,CRMConnector.I_ENTERTAINMENT,CRMConnector.I_FINANCE,CRMConnector↙
            .I_GOVERNMENT,CRMConnector.I_HEALTHCARE,CRMConnector.I_MANUFACTURING,         ↙
            CRMConnector.I_RETAIL};
        Go[] industryGoNames = {new Go(industryHREFNames[0],true,Go.METHOD_GET),new Go    ↙
            (industryHREFNames[1],true,Go.METHOD_GET),new Go(industryHREFNames[2],true,Go↙
            .METHOD_GET),new Go(industryHREFNames[3],true,Go.METHOD_GET),new Go           ↙
            (industryHREFNames[4],true,Go.METHOD_GET),new Go(industryHREFNames[5],true,Go↙
            .METHOD_GET),new Go(industryHREFNames[6],true,Go.METHOD_GET),new Go           ↙
```

```
                (industryHREFNames[7],true,Go.METHOD_GET)};
            String [] industryAnchorDisplayText                                    ↙
                = {"Advertising","Consulting","Entertainment","Finance","Government","Healthc↙
                are","Manufacturing","Retail"};
            com.thinairapps.tag.wml.Anchor [] industryAnchorNames ={new com.thinairapps.tag. ↙
                wml.Anchor(industryGoNames[0],new Text(industryAnchorDisplayText[0])),new com↙
                .thinairapps.tag.wml.Anchor(industryGoNames[1],new Text                    ↙
                (industryAnchorDisplayText[1])),new com.thinairapps.tag.wml.Anchor         ↙
                (industryGoNames[2],new Text(industryAnchorDisplayText[2])),new com.       ↙
                thinairapps.tag.wml.Anchor(industryGoNames[3],new Text                     ↙
                (industryAnchorDisplayText[3])),new com.thinairapps.tag.wml.Anchor         ↙
                (industryGoNames[4],new Text(industryAnchorDisplayText[4])),new com.       ↙
                thinairapps.tag.wml.Anchor(industryGoNames[5],new Text                     ↙
                (industryAnchorDisplayText[5])),new com.thinairapps.tag.wml.Anchor         ↙
                (industryGoNames[6],new Text(industryAnchorDisplayText[6])),new com.       ↙
                thinairapps.tag.wml.Anchor(industryGoNames[7],new Text                     ↙
                (industryAnchorDisplayText[7]))};
            int i;
            for (i = 0; i < 8; i++)
            {
                industryHREFNames[i] = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD   ↙
                    + "=" + industryURLParams[i]+ "&amp;rnd=" + Math.random();
                industryGoNames[i] = new Go(industryHREFNames[i],true,Go.METHOD_GET);
                if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
                {
                    p.addChild(new Break());
                    industryAnchorNames[i] = new Anchor(industryGoNames[i], new Text       ↙
                        (industryAnchorDisplayText[i]));
                    p.addChild(industryAnchorNames[i]);
                    p.addChild(new Break());
                }
                else
                {
                    industryAnchorNames[i] = new Anchor(industryGoNames[i], new Text       ↙
                        (industryAnchorDisplayText[i]));
                    p.addChild(industryAnchorNames[i]);
                }
            }

            //add the second Paragraph to the card
            card1.addParagraph(p);

            //add the card to the deck
            deck.addCard(card1);
        }
        else if (view.equals (CRMConnector.VIEW_BY_SALESCONTACT))
        {
            Bold b = new Bold(new Text("View By Sales Contact:"));
            p.addChild(b);
            p.addChild(new Break());

            //add the Paragraph to the card
            card1.addParagraph(p);

            p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

            //Add links to all the possible actions
            String [] salescontactHREFNames                                             ↙
                = {"arHref","jfHref","ltHref","myHref","mbHref","ndHref","rfHref","sdHref"};
            String [] salescontactURLParams = {CRMConnector.SC_ARTHUR_RIMBAUD,CRMConnector.  ↙
                SC_JOE_FRAZIER,CRMConnector.SC_LEON_TROTSKY,CRMConnector.SC_MICHELLE_YEOH,   ↙
                CRMConnector.SC_MIKHAIL_BULGAKOV,CRMConnector.SC_NEIL_DIAMOND,CRMConnector.  ↙
                SC_RICHARD_FEYNMAN,CRMConnector.SC_SAM_DONALDSON};
            Go[] salescontactGoNames = {new Go(salescontactHREFNames[0],true,Go.METHOD_GET), ↙
                new Go(salescontactHREFNames[1],true,Go.METHOD_GET),new Go                   ↙
                (salescontactHREFNames[2],true,Go.METHOD_GET),new Go(salescontactHREFNames   ↙
                [3],true,Go.METHOD_GET),new Go(salescontactHREFNames[4],true,Go.METHOD_GET), ↙
                new Go(salescontactHREFNames[5],true,Go.METHOD_GET),new Go                   ↙
                (salescontactHREFNames[6],true,Go.METHOD_GET),new Go(salescontactHREFNames   ↙
                [7],true,Go.METHOD_GET)};
            String [] salescontactAnchorDisplayText = {"Arthur Rimbaud","Joe Frazier","Leon ↙
                Trotsky","Michelle Yeoh","Mikhail Bulgakov","Neil Diamond","Richard ↙
                Feynman","Sam Donaldson"};
            com.thinairapps.tag.wml.Anchor [] salescontactAnchorNames ={new com.thinairapps. ↙
                tag.wml.Anchor(salescontactGoNames[0],new Text(salescontactAnchorDisplayText ↙
                [0])),new com.thinairapps.tag.wml.Anchor(salescontactGoNames[1],new Text ↙
                (salescontactAnchorDisplayText[1])),new com.thinairapps.tag.wml.Anchor ↙
                (salescontactGoNames[2],new Text(salescontactAnchorDisplayText[2])),new com. ↙
                thinairapps.tag.wml.Anchor(salescontactGoNames[3],new Text ↙
                (salescontactAnchorDisplayText[3])),new com.thinairapps.tag.wml.Anchor ↙
                (salescontactGoNames[4],new Text(salescontactAnchorDisplayText[4])),new com. ↙
                thinairapps.tag.wml.Anchor(salescontactGoNames[5],new Text ↙
                (salescontactAnchorDisplayText[5])),new com.thinairapps.tag.wml.Anchor ↙
                (salescontactGoNames[6],new Text(salescontactAnchorDisplayText[6])),new com. ↙
                thinairapps.tag.wml.Anchor(salescontactGoNames[7],new Text ↙
                (salescontactAnchorDisplayText[7]))};
            int i;
            for (i = 0; i < 8; i++)
            {
                salescontactHREFNames[i] = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD↙
                    + "=" + salescontactURLParams[i]+ "&amp;rnd=" + Math.random();
                salescontactGoNames[i] = new Go(salescontactHREFNames[i],true,Go.METHOD_GET);
                if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
                {
                    p.addChild(new Break());
                    salescontactAnchorNames[i] = new Anchor(salescontactGoNames[i], new Text ↙
                        (salescontactAnchorDisplayText[i]));
                    p.addChild(salescontactAnchorNames[i]);
                    p.addChild(new Break());
                }
                else
                {
                    salescontactAnchorNames[i] = new Anchor(salescontactGoNames[i], new Text ↙
                        (salescontactAnchorDisplayText[i]));
                    p.addChild(salescontactAnchorNames[i]);
                }
            }

            //add the second Paragraph to the card
            card1.addParagraph(p);

            //add the card to the deck
            deck.addCard(card1);
        }
        String resultString = deck.render();
        return resultString;
    }


    /**
     *A user selects a field value by which to sort the contents of the folder and
     * then this method is called to display all the items that have that value
     *
     * @param fieldValue The value of the field by which the user wants to sort the folder
     * @param access A handle to ConnectorAccess and the ThinAir Server services
     * @param sessionId An identifier of the user's already established session
     * @return A collection of StoreItems that satisfy the criteria of the user
     */
    static String viewByField(String fieldValue, ConnectorAccess access, String sessionId) ↙
        throws Exception
    {
        //Create the deck, and add a few elements to it
        WMLTagDocument deck = new WMLTagDocument();
        String url = null;
        DisplayCard card = new DisplayCard("c1");
        Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);
        Bold b = new Bold(new Text("Matching Items:"));
        p.addChild(b);
        p.addChild(new Break());
        card.addParagraph (p);
        Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT,Paragraph.MODE_NOWRAP);

        //The cache for this session
        Hashtable cache = null;

        //The Vector that will store the items that have fields that match the incoming
        //field parameter
        Vector itemswMatchingfields = new Vector(15);

        //Get the cache for this session
        cache = access.getSessionCache(sessionId);

        //Retrieve the Store Items that we've already placed into the cache
        StoreItems customItems = ((StoreItems)cache.get("storeitems"));

        //we get an Enumeration of the items...
        Enumeration sortedItemEnum = (((Vector)customItems).elements());

        boolean didAnyItemsMatch = false;
        //Go through the items, and identify those that have the field that
        //has been passed in as the search parameter.
        int itemIterated = 0;
        String elementNumber = "elemnum";
        String href = "";
        Go go = new Go (href, true, Go.METHOD_GET);
        Anchor itemAnchor;
        while (sortedItemEnum.hasMoreElements())
        {
            String fieldText = null;
            String companyName = null;
            CustomItem custItem = (CustomItem)sortedItemEnum.nextElement();

            //Get the fields of the item
            Data customFields = custItem.getCustomFieldData();

            //Get an enumeration of the fields
            Enumeration fieldEnum = customFields.getFields();

            //If the search parameter matches a field on the item, then we return a link to ↙
                the
            //item with the item's company name displayed.
            boolean hasField = false;
            while (fieldEnum.hasMoreElements())
            {
                Field thisField = (Field)fieldEnum.nextElement();
                //Get the Item's Company Name field.  We need it for displaying a link to the↙
                    item.
                if (thisField.getName().equals ("CompanyName"))
                {
                    companyName = thisField.getString();
                    if (hasField == true)
                    {
                        //Create the link to the Item, with the Company Name field rendered
                        href = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +     ↙
                            CRMConnector.VIEW_BY_FIELD_ACTION + "&amp;elemnum=" +             ↙
                            itemIterated + "&amp;rnd=" + Math.random();

                        go = new Go(href, true, Go.METHOD_GET);

                        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
                        {
                            itemAnchor = new Anchor (go, new Text (companyName));
                            p2.addChild(new Break());
                            p2.addChild(itemAnchor);
                        }
                    else
                    {
                        itemAnchor = new Anchor (go, new Text (companyName));
                        p2.addChild(itemAnchor);
                    }

                    p2.addChild(new Break());
                    didAnyItemsMatch = true;

                    companyName = null;
                    hasField = false;
                    break;
                }
                else
                    continue;
            }

            //We will display links only to those items that match the search criteria
            else if (thisField.getString().equals(fieldValue))
            {
                hasField = true;
                if (!(companyName == null))
                {
                    //Create the link to the Item, with the Company Name field rendered
                    href = CRMConnector.path+ "?" + CRMConnector.ACTION_FIELD + "=" +      ↙
                        CRMConnector.VIEW_BY_FIELD_ACTION + "&amp;elemnum=" +             ↙
                        itemIterated + "&amp;rnd=" + Math.random();

                    go = new Go(href, true, Go.METHOD_GET);

                    if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
                    {
                        itemAnchor = new Anchor (go, new Text (companyName));
                        p2.addChild(new Break());
                        p2.addChild(itemAnchor);
                    }
                    else
                    {
                        itemAnchor = new Anchor (go, new Text (companyName));
                        p2.addChild(itemAnchor);
                    }

                    p2.addChild(new Break());
                    didAnyItemsMatch = true;

                    companyName = null;
                    hasField = false;
                    break;
                }
                else
                    continue;
            }
        } // end while
    itemIterated++;
    }

    //If no items matched the criteria, render this fact
    if (didAnyItemsMatch == false)
    {
        p2.addChild(new Break());
        p2.addChild(new Text("No Items to Display"));
        p2.addChild(new Break());
        p2.addChild(new Break());
    }
    //Else add one more break
    else if (!(didAnyItemsMatch == false))
    {
        p2.addChild(new Break());
    }
```

```
        //link home.
        String startHref = CRMConnector.path+ "?rnd="+Math.random();

        Go startGo = new Go(startHref,true,Go.METHOD_GET);
        Anchor startaAnchor;

        if (CRMConnector.g_DEVICE instanceof NokiaWAPDevice)
        {
            p2.addChild(new Break());
            startaAnchor = new Anchor(startGo,new Text("Start again..."));
            p2.addChild(startaAnchor);
        }
        else
        {
            startaAnchor = new Anchor(startGo,new Text("Start again..."));
            p2.addChild(startaAnchor);
        }
        card.addParagraph(p2);
        deck.addCard (card);
        return deck.render();
    }


    /**
     * Renders an input form with the values preset
     *
     * Pass in the field values that the item had.
     * @param item --
     * @param messageID --
     *
     */
    static String editItem(CustomItem item, String messageID)
    {
    //Get the Data object that contains all our custom fields.
    Data customFields = item.getCustomFieldData();

    //Get the fields that we're expecting
    String customerNameField = customFields.getField("CustomerName").valueToString();
    String positionField = customFields.getField("Position").valueToString();
    String companyNameField = customFields.getField("CompanyName").valueToString();
    String industryField = customFields.getField("Industry").valueToString();
    String itemCreatedField = customFields.getField("ItemCreated").valueToString();
    String salesContactField = customFields.getField("SalesContact").valueToString();
    String accountNumberField = customFields.getField("AccountNumber").valueToString();
    String customerStatusField = customFields.getField("CustomerStatus").valueToString();

    //Create the deck
    WMLTagDocument deck = new WMLTagDocument();

    //Create a MultipleInputCard
    MultipleInputCard card1 = new MultipleInputCard("g1");

    //Allow the user to type in information and set the input text to lowercase by default
    LabeledInput custName = new LabeledInput("cstnm","Customer Name:");
    custName.setFormat("*m");
    //Set the default value
    custName.addAttribute ("value", customerNameField);

    LabeledInput position = new LabeledInput("psn","Position:");
    position.setFormat("*m");
    //Set the default value
    position.addAttribute ("value", positionField);

    LabeledInput compName = new LabeledInput("cnm","Company Name:");
    compName.setFormat("*m");
    //Set the default value
    compName.addAttribute ("value", companyNameField);
    LabeledInput[] inputs1 = {custName, position, compName};

    //A link to the second card
    card1.buildCard("#g2","OK",inputs1, Go.METHOD_GET);
    deck.addCard(card1);

    //Create a select card
    Card card2 = new Card ("g2", "industry");

    //Create the Do
    Do doElem = new Do(Do.TYPE_ACCEPT,new Go("#g3",false));

    //Add the do
    card2.addChild(doElem);

    //Create a paragraph
    Paragraph p = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);
    p.addChild(new Text("Industry:"));

    //Add the paragraph
    card2.addParagraph(p);

    //Add another paragraph
    Paragraph p2 = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);

    //Create a select
    Select industrySelect = new Select("","industry",false);

    String[] industryOptionNames                                                        ↙
        = {"Advertising","Consulting","Entertainment","Finance","Government","Health    ↙
        Care","Manufacturing","Retail"};
    Option[] industryOptions = new Option [8];
    String[] industryOptionValues = {"a","b","c","d","e","f","g","h"};
    String[] industryOptionIValues = {"1","2","3","4","5","6","7","8"};

    int i;
    for (i=0; i<8; i++)
    {
        industryOptions[i] = new Option ("OK",industryOptionValues[i],industryOptionNames    ↙
            [i]);
        industrySelect.addOption(industryOptions[i]);
        if (industryField.equals (industryOptionValues[i]))
            industrySelect.setINameAndIValue (industryOptionValues[i],industryOptionIValues  ↙
                [i]);
    }

    //Add the select to the paragraph
    p2.addChild(industrySelect);

    //Add the second paragraph
    card2.addChild(p2);

    //Add the card
    deck.addCard(card2);


    //Create a select for Sales Contact
    Card card5 = new Card ("g3");

    //Create the Do
    Do salesDo = new Do(Do.TYPE_ACCEPT,new Go("#g4",false));

    //Add the do
    card5.addChild(salesDo);

    //Create a paragraph
    Paragraph sP = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);
    sP.addChild(new Text("Sales Contact:"));
    //Add the paragraph
    card5.addParagraph(sP);

    //Add another paragraph
    Paragraph sP2 = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);

    //Create a select
    Select salesSelect = new Select("","salesContact",false);

    String[] salesOptionNames = {"Mikhail Bulgakov","Neil Diamond","Sam Donaldson","Richard ↙
        Feynman","Joe Frazier","Arthur Rimbaud","Leon Trotsky","Michelle Yeoh"};
    Option[] salesOptions = new Option [8];
    String[] salesOptionValues = {"a","b","c","d","e","f","g","h"};
    String[] salesOptionIValues = {"1","2","3","4","5","6","7","8"};

    int k;
    for (k=0; k<8; k++)
    {
        salesOptions[k] = new Option ("OK",salesOptionValues[k],salesOptionNames[k]);
        salesSelect.addOption(salesOptions[k]);
        if (salesContactField.equals (salesOptionValues[k]))
            salesSelect.setINameAndIValue (salesOptionValues[k],salesOptionIValues[k]);
    }

    //Add the select to the paragraph
    sP2.addChild(salesSelect);

    //Add the second paragraph
    card5.addChild(sP2);

    //Add the card
    deck.addCard(card5);

    //Create a MultipleInputCard
    MultipleInputCard card3 = new MultipleInputCard("g4");

    LabeledInput actNumber = new LabeledInput("an","Account Number:");
    actNumber.setFormat("*m");
    actNumber.addAttribute("value",accountNumberField);

    LabeledInput[] inputs2 = {actNumber};

    card3.buildCard("#g5","OK",inputs2, Go.METHOD_GET);
    deck.addCard(card3);


    //Create a select card
    Card card6 = new Card ("g5", "Customer Status");

    //Create a paragraph
    Paragraph p3 = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);
    p3.addChild(new Text("Customer Status"));

    //Add the paragraph
    card6.addParagraph(p3);

    //Add another paragraph
    Paragraph p4 = new Paragraph(Paragraph.ALIGN_LEFT, Paragraph.MODE_NOWRAP);

    //Create a select
    Select custstatusSelect = new Select("","custstatus",false);

    String[] custstatusOptionNames = {"Needs First Contact","Needs Follow-Up","Needs Credit ↙
        Approval","Needs to be Invoiced","Credit Approved","Invoice Sent","Credit         ↙
        Denied","Dead End"};
    Option[] custstatusOptions = new Option [8];
    String[] custstatusOptionValues = {"a","b","c","d","e","f","g","h"};
    String[] custstatusOptionIValues = {"1","2","3","4","5","6","7","8"};
    int j;
    for (j=0; j<8; j++)
    {
        custstatusOptions[j] = new Option ("OK",custstatusOptionValues[j],          ↙
            custstatusOptionNames[j]);
        custstatusSelect.addOption(custstatusOptions[j]);
        if (customerStatusField.equals (custstatusOptionValues[j]))
            custstatusSelect.setINameAndIValue (custstatusOptionValues[j],          ↙
                custstatusOptionIValues[j]);
    }

    //Set the URL params to the values in the WML variables
    //&amp;, the escape sequence for ampersand, delimits name-value pairs. $ is used to   ↙
        dereference a WML variable.
    String href;
    href = CRMConnector.path + "?action=" + CRMConnector.UPDATE_ACTION + "&amp;"+"MessageID  ↙
        ="+messageID+"&amp;cstnm=$cstnm&amp;psn=$psn&amp;cnm=$cnm&amp;industry=$industry&amp;↙
        spm=$spm&amp;sc=$salesContact&amp;an=$an&amp;custstatus=$custstatus&amp;rnd="+Math.  ↙
        random();

    //Create the Do
    Do custDo = new Do(Do.TYPE_ACCEPT,new Go(href,false));

    //Add the do
    card6.addChild(custDo);

    //Add the select to the paragraph
    p4.addChild(custstatusSelect);

    //Add the second paragraph
    card6.addChild(p4);

    //Add the card
    deck.addCard(card6);

    //Render the deck
    return deck.render();
    }
}//end Connector
```

SCANNED,# 8

Application deficiencies were found during scanning:

☐ Page(s)__________ of drawings were not present for scanning.
(Document title)

☐ Page(s)__________ of ____________________ were not present for scanning.
(Document title)

☐ Scanned copy is best available.